iamneo

AGILE

Agility

Agility has become today's buzzword when describing a modern software process.

An agile team is a nimble team able to appropriately respond to changes. Change is what software development is very much about.

Changes in the software being built, changes to the team members, changes because of new technology, changes of all kinds that may have an impact on the product they build or the project that creates the product.

An agile team recognizes that software is developed by individuals working in teams and that the skills of these people, their ability to collaborate is at the core for the success of the project.

Agility

To accomplish agility to any software process,

1. It is essential that the process be designed in a way that allows the project team to adapt tasks and to streamline them
2. Conduct planning in a way that understands the fluidity of an agile development approach
3. Eliminate all but the most essential work products and keep them lean
4. Emphasize an incremental delivery strategy that gets working software to the customer as rapidly as feasible for the product type and operational environment.

Agile Process - Agility Principles

The Agile Alliance defines 12 agility principles for those who want to achieve agility

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

Agile Process - Agility Principles

The Agile Alliance defines 12 agility principles for those who want to achieve agility

5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7	Working software is the primary measure of progress.
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Agile Process - Agility Principles

The Agile Alliance defines 12 agility principles for those who want to achieve agility

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.

Agile Process - Human Factors

Agile development focuses on the talents and skills of individuals, molding the process to specific people and teams.

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

Agile Process - Human Factors

Agile Process - Human Factors

Competence

In an agile development 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.

Agile Process - Human Factors

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.

Agile Process - Human Factors

Software engineering 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 that provides business value for the customer.

To accomplish these tasks, team members must collaborate with one another and all other stakeholders.

Collaboration

Agile Process - Human Factors

Any good software team 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.

Decision Making Ability

Agile Process - Human Factors

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.

Fuzzy Problem Solving Ability

Agile Process - Human Factors

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.‖

Agile Process - Human Factors

Self Organization

Self-organization implies three things

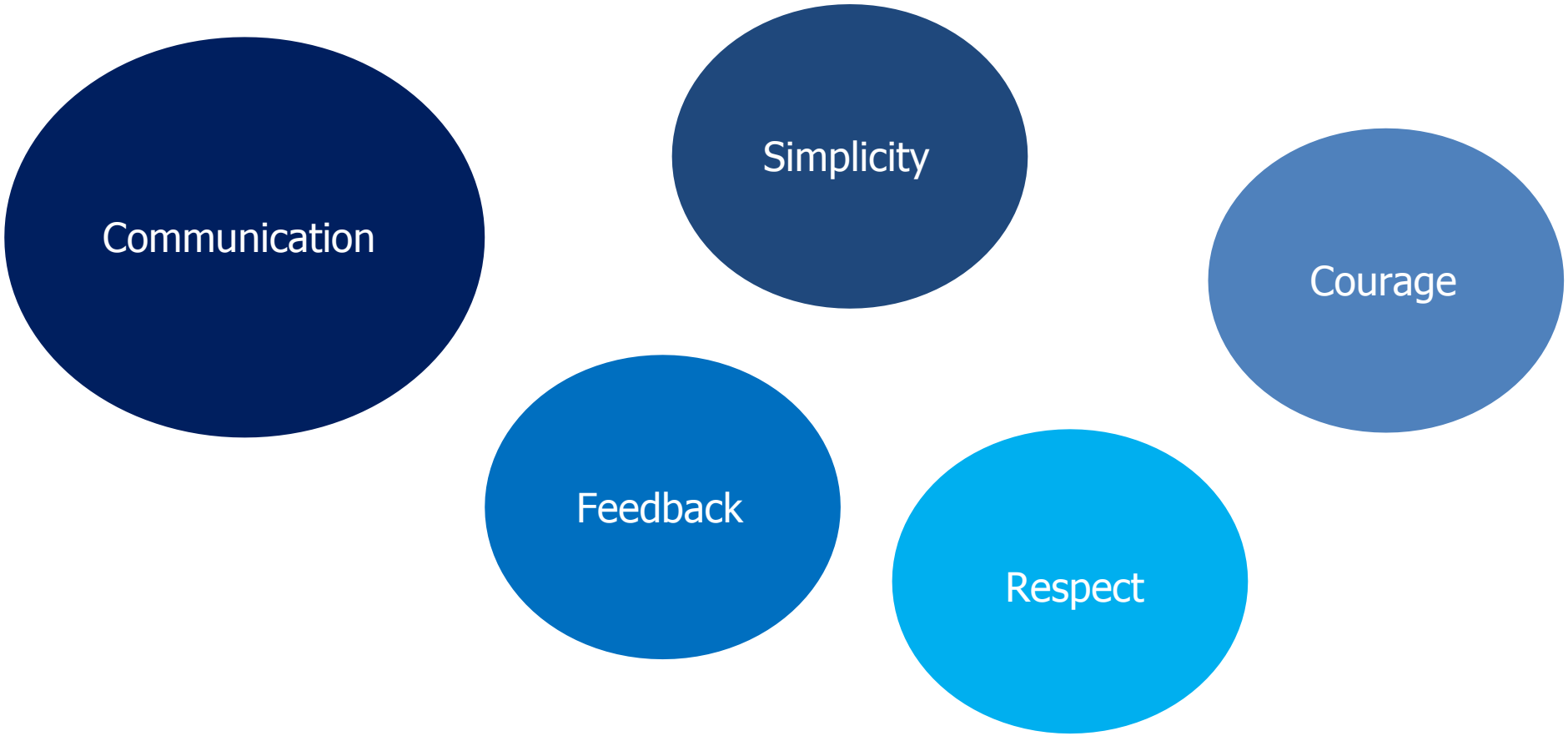
The agile team organizes itself for the work to be done

The team organizes the process to best accommodate its local environment

The team organizes the work schedule to best achieve delivery of the software increment

Extreme Programming - Values

Five values that establish a foundation for all work performed as part of XP



Extreme Programming - Values

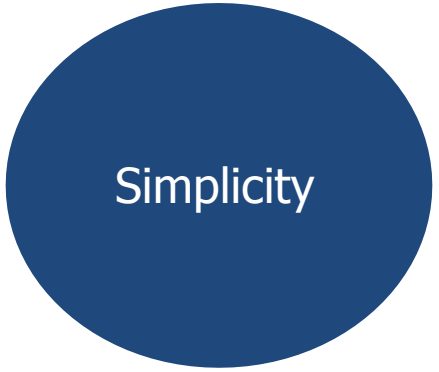
In order to achieve effective communication between software engineers and other stakeholders, XP emphasizes close, yet informal (verbal) collaboration between customers and developers, the establishment of effective metaphors for communicating important concepts, continuous feedback, and the avoidance of voluminous documentation as a communication medium.

Extreme Programming - Values

To achieve simplicity, XP restricts developers to design only for immediate needs, rather than consider future needs.

The intent is to create a simple design that can be easily implemented in code.

If the design must be improved, it can be refactored at a later time.



Extreme Programming - Values

Feedback is derived from three sources: the implemented software itself, the customer, and other software team members.

By designing and implementing an effective testing strategy, the software provides the agile team with feedback. XP makes use of the unit test as its primary testing tactic.

Finally, as new requirements are derived as part of iterative planning, the team provides the customer with rapid feedback regarding cost and schedule impact.

Extreme Programming - Values

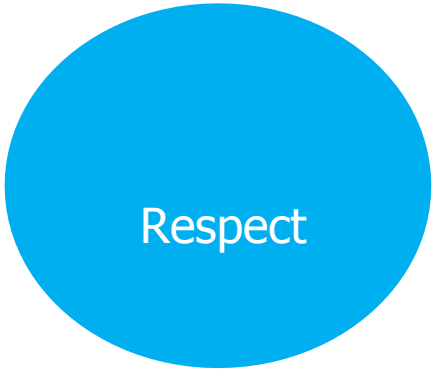
Strict adherence to certain XP practices demands courage. A better word might be discipline.

An agile XP team must have the discipline to design for today, recognizing that future requirements may change dramatically, thereby demanding substantial rework of the design and implemented code.

Extreme Programming - Values

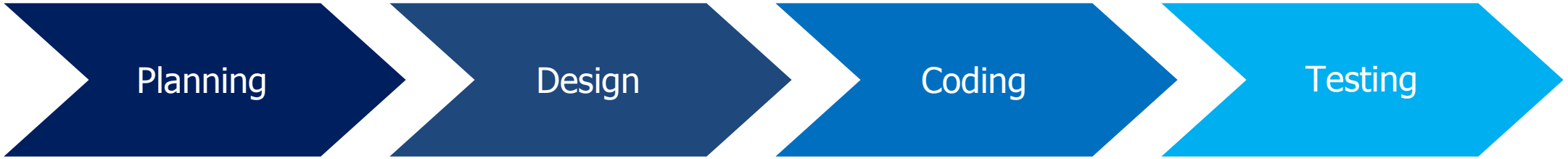
By following each of these values, the agile team inculcates respect among it members, between other stakeholders and team members, and indirectly, for the software itself.

As they achieve successful delivery of software increments, the team develops growing respect for the XP process.



Extreme Programming - Process

Extreme Programming uses an object-oriented approach as its preferred development paradigm and encompasses a set of rules and practices that occur within the context of four framework activities



Extreme Programming - Process

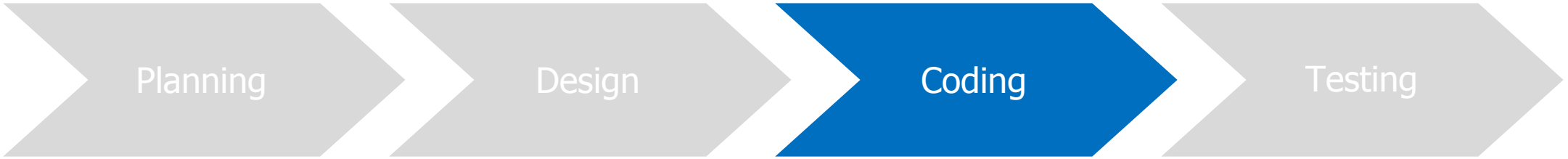
The planning activity begins with listening a requirements gathering activity that enables the technical members of the XP team to understand the business context for the software and to get a broad feel for required output and major features and functionality.

Extreme Programming - Process

XP design rigorously follows the KIS (keep it simple) principle. A simple design is always preferred over a more complex representation. In addition, the design provides implementation guidance for a story as it is written nothing less, nothing more.

Extreme Programming - Process

After stories are developed and preliminary design work is done, the team does not move to code, but rather develops a series of unit tests that will exercise each of the stories that is to be included in the current release. Once the unit test has been created, the developer is better able to focus on what must be implemented to pass the test.



Extreme Programming - Process

As the individual unit tests are organized into a ―universal testing suite‖, integration and validation testing of the system can occur on a daily basis. This provides the XP team with a continual indication of progress and also can raise warning flags early if things go wrong.

XP acceptance tests, also called customer tests, are specified by the customer and focus on overall system features and functionality that are visible and reviewable by the customer.

Extreme Programming - Industrial XP

IXP incorporates six new practices that are designed to help ensure that an XP project works successfully for significant projects within a large organization.

1. Readiness Assessment
2. Project Community
3. Project Chartering
4. Test-driven Management
5. Retrospectives
6. Continuous learning

Extreme Programming - Industrial XP

1. Readiness Assessment

- The assessment ascertains whether

1. an appropriate development environment exists to support IXP
2. the team will be populated by the proper set of stakeholders
3. the organization has a distinct quality program and supports continuous improvement
4. the organizational culture will support the new values of an agile team
5. the broader project community will be populated appropriately.

Extreme Programming - Industrial XP

2. Project Community

A community may have a technologist and customers who are central to the success of a project as well as many other stakeholders who —are often at the periphery of an IXP project yet they may play important roles on the project‖.

In IXP, the community members and their roles should be explicitly defined and mechanisms for communication and coordination between community members should be established.

Extreme Programming - Industrial XP

3. Project Chartering

The IXP team assesses the project itself to determine whether an appropriate business justification for the project exists and whether the project will further the overall goals and objectives of the organization.

Chartering also examines the context of the project to determine how it complements, extends, or replaces existing systems or processes.

Extreme Programming - Industrial XP

4. Test-driven Management

An IXP project requires measurable criteria for assessing the state of the project and the progress that has been made to date.

Test-driven management establishes a series of measurable ―destinations‖ and then defines mechanisms for determining whether or not these destinations have been reached.

Extreme Programming - Industrial XP

5. Retrospectives

An IXP team conducts a specialized technical review after a software increment is delivered. Called a retrospective, the review examines —issues, events, and lessons-learned‖ across a software increment and/or the entire software release. The intent is to improve the IXP process.

Extreme Programming - Industrial XP

6. Continuous learning

Because learning is a vital part of continuous process improvement, members of the XP team are encouraged to learn new methods and techniques that can lead to a higher quality product.

Extreme Programming - Issues

- Requirements volatility
- Conflicting customer needs
- Requirements are expressed informally
- Lack of formal design

Extreme Programming - Issues

Requirements volatility

Conflicting customer needs

Requirements are expressed informally

Lack of formal design

Because the customer is an active member of the XP team, changes to requirements are requested informally.

As a consequence, the scope of the project can change and earlier work may have to be modified to accommodate current needs.

Proponents argue that this happens regardless of the process that is applied and that XP provides mechanisms for controlling scope creep.

Extreme Programming - Issues

Requirements volatility

Conflicting customer needs

Requirements are expressed informally

Lack of formal design

Many projects have multiple customers, each with his own set of needs.

In XP, the team itself is tasked with assimilating the needs of different customers, a job that may be beyond their scope of authority.

Extreme Programming - Issues

Requirements volatility

Conflicting customer needs

Requirements are expressed informally

Lack of formal design

User stories and acceptance tests are the only explicit manifestation of requirements in XP.

Critics argue that a more formal model or specification is often needed to ensure that omissions, inconsistencies, and errors are uncovered before the system is built.

Proponents counter that the changing nature of requirements makes such models and specification obsolete almost as soon as they are developed.

Extreme Programming - Issues

Requirements volatility

Conflicting customer needs

Requirements are expressed informally

Lack of formal design

XP deemphasizes the need for architectural design and in many instances, suggests that design of all kinds should be relatively informal.

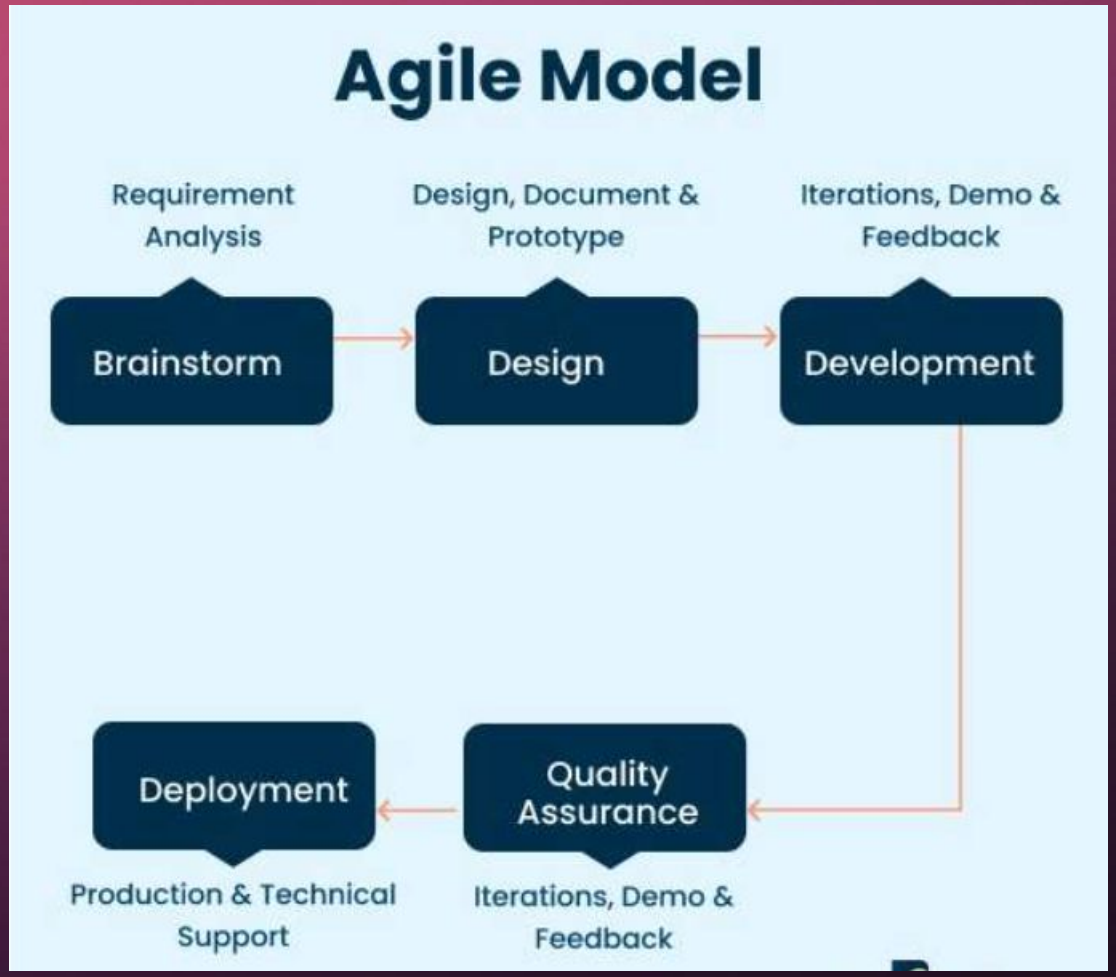
Critics argue that when complex systems are built, design must be emphasized to ensure that the overall structure of the software will exhibit quality and maintainability.

XP proponents suggest that the incremental nature of the XP process limits complexity and therefore reduces the need for extensive design.

Agile Model

iamneo

There are some common Software Life Cycle Examples or SDLC models available. They are as given below.



- A major characteristics of the Agile Model is that it demarcates the product into different cycles and produces a working product quickly.
- After every releases are tested, that back information is used in the following version for more improvement.
- There is a drawback in this model and that revolves around the dependency on customer interaction.
- The Agile leader gives complete support through out the processing of the model to the team

When to use Agile Model?

iamneo

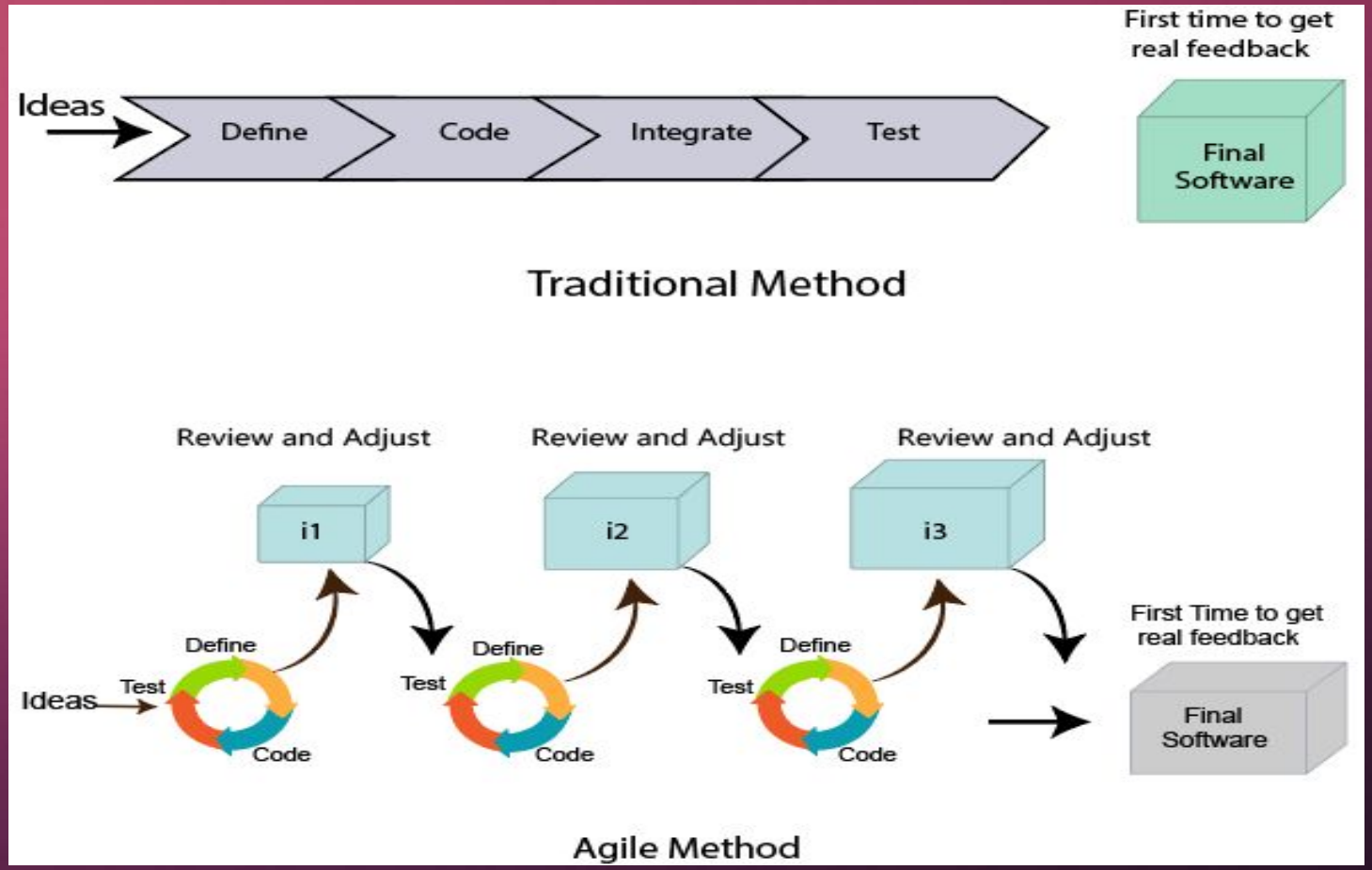
- To implement new changes
- To implement a feature that would require some extra hours
- Very few initial requirements
- Product owner involvement is high

Agile Methodology

iamneo

- An agile methodology is an iterative approach to software development.
- Each iteration of agile methodology takes a short time interval of 1 to 4 weeks.
- The agile development process is aligned to deliver the changing business requirement.
- It distributes the software with faster and fewer changes.
- The single-phase software development takes 6 to 18 months.
- In single-phase development, all the requirement gathering and risks management factors are predicted initially.

Agile Methodology



Agile Development Model Phases

iamneo

1.Requirements gathering

2.Design the requirements

3.Construction/ iteration

4.Testing/ Quality assurance

5.Deployment

6.Feedback

Requirements gathering:

In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

Design the requirements:

When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system

Construction/ iteration:

When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.

Testing:

In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

Deployment:

In this phase, the team issues a product for the user's work environment.

Feedback:

After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

iamneo

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- Extreme Programming(XP)

Agile Manifesto:

In February 2001, at the Snowbird resort in Utah, a team of 17 software developers met to discuss lightweight development methods. The result of their meeting was the following Agile Manifesto for software development

- ✔Individuals and interactions over Processes and tools.
- ✔Working software over comprehensive documentation.
- ✔Customers are collaboration over contact negotiation.
- ✔Responding to change over following a plan.

Principles of Agile

iamneo

Customer Satisfaction:

Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.

Welcome Change:

Making changes during software development is common and inevitable. Every changing requirement should be welcome, even in the late development phase. Agile process works to increase the customers' competitive advantage

Deliver the Working Software:

Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest time period.

Collaboration:

Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.

Motivation:

Projects should be build around motivated team members. Provide such environment that supports individual team members and trust them. It makes them feel responsible for getting the job done thoroughly.

Face-to-face Conversation:

Face-to-face conversation between Scrum Master(Professional leading the team) and development team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.

Measure the Progress as per the Working Software:

The working software is the key and primary measure of the progress.

Maintain Constant Pace:

The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.

Monitoring:

Pay regular attention to technical excellence and good design to maximize agility.

Simplicity:

Keep things simple and use simple terms to measure the work that is not completed.

Self-organized Teams:

The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams

Review the Work Regularly:

The work should be reviewed at regular intervals, so that the team can reflect on how to become more productive and adjust its behavior accordingly.

Scrum

iamneo

Scrum is a framework that helps teams work together.

Scrum encourages teams to learn through experiences, self organize while working on a problem and reflect on their wins and losses to continuously improve. The Development team attends the scrum meeting on regular basis.

Scrum is structured to adapt team to naturally adapt to changing conditions and user requirements, with re-prioritization built into the process and short release cycles so your team can constantly learn and improve.

The main responsibilities of the scrum master is removing impediments, facilitating meeting as and when requested and consulting the devel t team and product Owner

Burn Down Chart

iamneo

A burn down chart is a graphical representation of work left to do versus time.

It can be applied to any project containing measurable progress over time

In a burn down chart the outstanding work is often on the vertical axis, with time along the horizontal.

It is useful for predicting when all of the work will be completed.

In the daily scrum, the Development team updates the Sprint Burn Down and plots the remaining work of the day.

Reasons for Burn Down Chart:

1. Monitoring the project scope creep
2. Keeping the team running on schedule
3. Comparing the planned work against the team progression

Crystal

iamneo

The crystal method is an agile methodology that is considered a lightweight that focuses on individuals and their interactions.

It is mainly for short term projects by a team of developers working out of a single workspace.

Two Core benefits of Crystal Method:

1. Find your own way and methods to optimize workflow
2. Make use of unique methods to make the project unique and dynamic

Properties of Crystal Framework

iamneo

1. Frequent Delivery
2. Reflective Improvement
3. Osmotic communication – team picking up relevant information
4. Personal Safety
5. Focus
6. Easy access to expert users
7. Technical tooling – required tools

How does Crystal Function

iamneo

1. Crystal Clear – short term projects – 1 to 6 members – single workspace
2. Crystal Yellow – small team size – 7 to 20 members – automated testing – resolve bugs – reduce documentations
3. Crystal Orange – 21 to 40 members – team splits as per the function – project last to 1 to 2 years
4. Crystal Orange Web – 21 to 40 members – project used by public – series requires programming
5. Crystal Red – 40 to 80 members – team formed and divided according to requirements
6. Crystal Maroon – large size project – 80 to 200 members – methods are different as per the requirement of the software
7. Crystal Diamond – Large Projects

Adaptive Software Development

iamneo

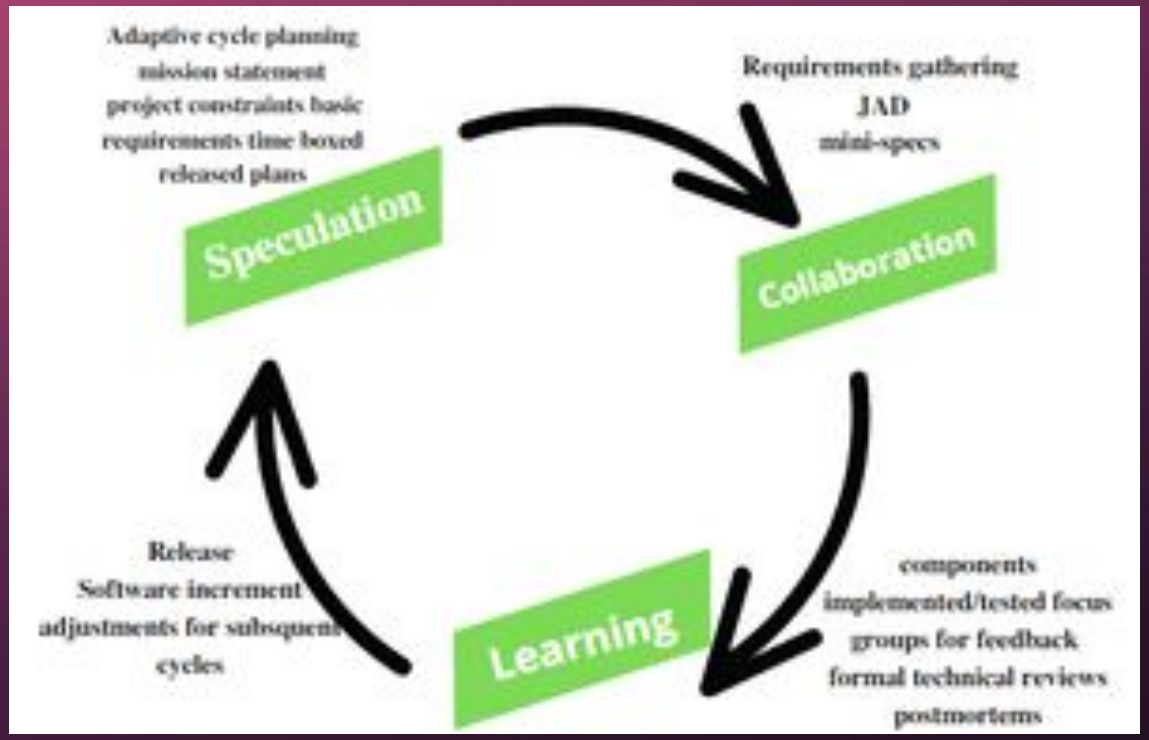
Adaptive Software Development is a method to build complex software and system.

ASD life cycle incorporates three phases

Speculation

Collaboration

Learning



Speculation:

During this phase the project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs, customer mission statement, etc. to define set of release cycles that the project wants.

Collaboration:

It collaborates communication and teamwork but emphasizes individualism as individual creativity play a major role in creative thinking.

Criticize without animosity

Assist without resentment

Work as hard as possible

Possession of skill set

Communicate problems to find effective solution

Learning:

Learning helps the workers to increase their level of understanding over the project.

Learning process is of 3 ways

Focus groups – group discussion

Technical reviews – feedback on the products

Project postmortem – feedback on continuous improvement on both current and future projects

Dynamic Software Development

iamneo

The DSD technique is an associate degree agile code development approach that provides a framework for building and maintaining systems.

The DSDM philosophy is borrowed from a modified version of the sociologist principle – 80% of An application is often delivered in twenty percent of the time it would desire deliver the entire application.

DSDM Life Cycle:

1. Feasibility Study
2. Business Study
3. Functional Model Iteration
4. Design and Build Iteration
5. Implementation

Feasibility Study:

It establishes the essential business necessities and constraints related to the applying to be designed then assesses whether or not the application could be viable candidate for the DSDM method.

Business Study:

It establishes the use and knowledge necessities that may permit the applying to supply business value; additionally, it is the essential application design and identifies the maintainability necessities for the applying.

Functional Model Iteration:

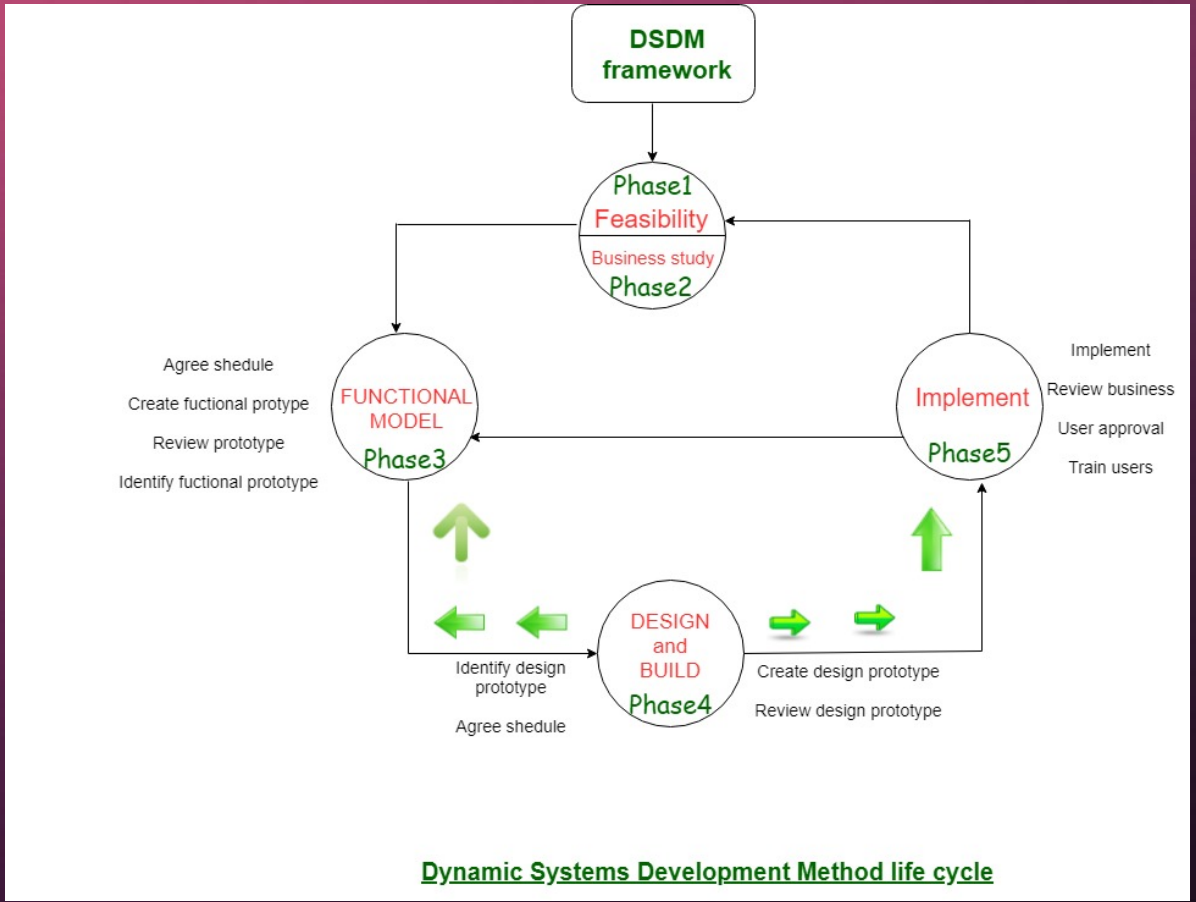
It produces a collection of progressive prototypes that demonstrates practically for the client.

Design and Build Iteration:

It revisits prototypes designed throughout useful model iteration to make sure that everyone has been designed during a manner that may alter it to supply operational business price for finish users.

Implementation:

It places the newest code increment into the operational surroundings.



Dynamic Systems Development Method life cycle

Feature Driven Development

iamneo

An Agile method of developing software, Feature-Driven Development (FDD) is customer – centric, iterative and incremental with the goal of delivering tangible software results often and efficiently.

FDD in Agile encourages status reporting at all levels, which helps track progress and results.

How does FDD Work?

1. Develop Overall Model
2. Build Feature List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

Lean Software Development (LSD)

iamneo

LSD is used to streamline and optimize the software development process.

It is also referred to as Minimum Viable Product (MVP)

As its intend is to speed up development by focusing on new delivarbles.

Key Principles of Lean Software Development:

1. Eliminating the waste
2. Fast Delivery
3. Amplify Learning
4. Builds Quality
5. Respect Teamwork
6. Delay the commitment
7. Optimizing the whole system

Advantages of LSD

iamneo

1. Removes unnecessary process stages when designing software so that it acts as a time saver as simplifies the development process.

2. With a focus on MVP, Lean Software Development prioritizes essential functions so this removes the risk of spending time on valueless builds.

3. It increases the involvement power of your team as more and more members participate due to which the overall workflow becomes optimized and losses get reduced.

AGILE UNIFIED PROCESS (AUP)

iamneo

The Agile Unified Process (AUP) adopts a "serial in the large" and "iterative in the small" philosophy for building computer-based systems.

Each AUP iteration addresses the following activities:

- **Modeling**. UML representations of the business and problem domains are created. However, to stay agile, these models should be "just barely good enough" to allow the team to proceed.
- **Implementation** Models are translated into source code.
- **Testing** Like XP, the team designs and executes a series of tests to uncover errors and ensure that the source code meets its requirements.
- **Deployment**. Like the generic process activity deployment in this context focuses on the delivery of a software increment and the acquisition of feedback from end users.
- **Configuration and project management.** In the context of AUP, configuration management addresses change management, risk management, and the control of any persistent work products that are produced by the team. Project management tracks and controls the progress of the team and coordinates team activities.
- **Environment management.** Environment management coordinates a process infrastructure that includes standards, tools, and other support technology available to the team.

A TOOL SET FOR THE AGILE PROCESS

iamneo

- Agile teams stress using tools that permit the rapid flow of understanding. Some of those tools are social, starting even at the hiring stage.
- Some tools are technological, helping distributed teams simulate
- being physically present. Many tools are physical, allowing people to manipulate them in workshops."
- Cockburn argues that "tools" that address these issues are critical success
- factors for agility. For example, a hiring "tool" might be the requirement to have a prospective team member spend a few hours pair programming with an existing member of the team. The "fit" can be assessed immediately.
- Collaborative and communication "tools" are generally low tech and incorporate any mechanism ("physical proximity, whiteboards, poster sheets, index cards, and sticky notes")
- Active communication is achieved via the team dynamics (e.g., pair programming), while passive communication is achieved by "information radiators" (e.g., a flat panel display that presents the overall status of different components of an increment).
- Project management tools deemphasize the Gantt chart and replace it with earned value charts or "graphs of tests created versus passed .
- Other agile tools are used to optimize the environment in which the agile team works (e.g., more efficient meeting areas), improve the team culture by nurturing social interactions(e.g., collocated teams), physical devices (e.g., electronic whiteboards), and process enhancement (e.g., pair programming or time-boxing)"