UPES

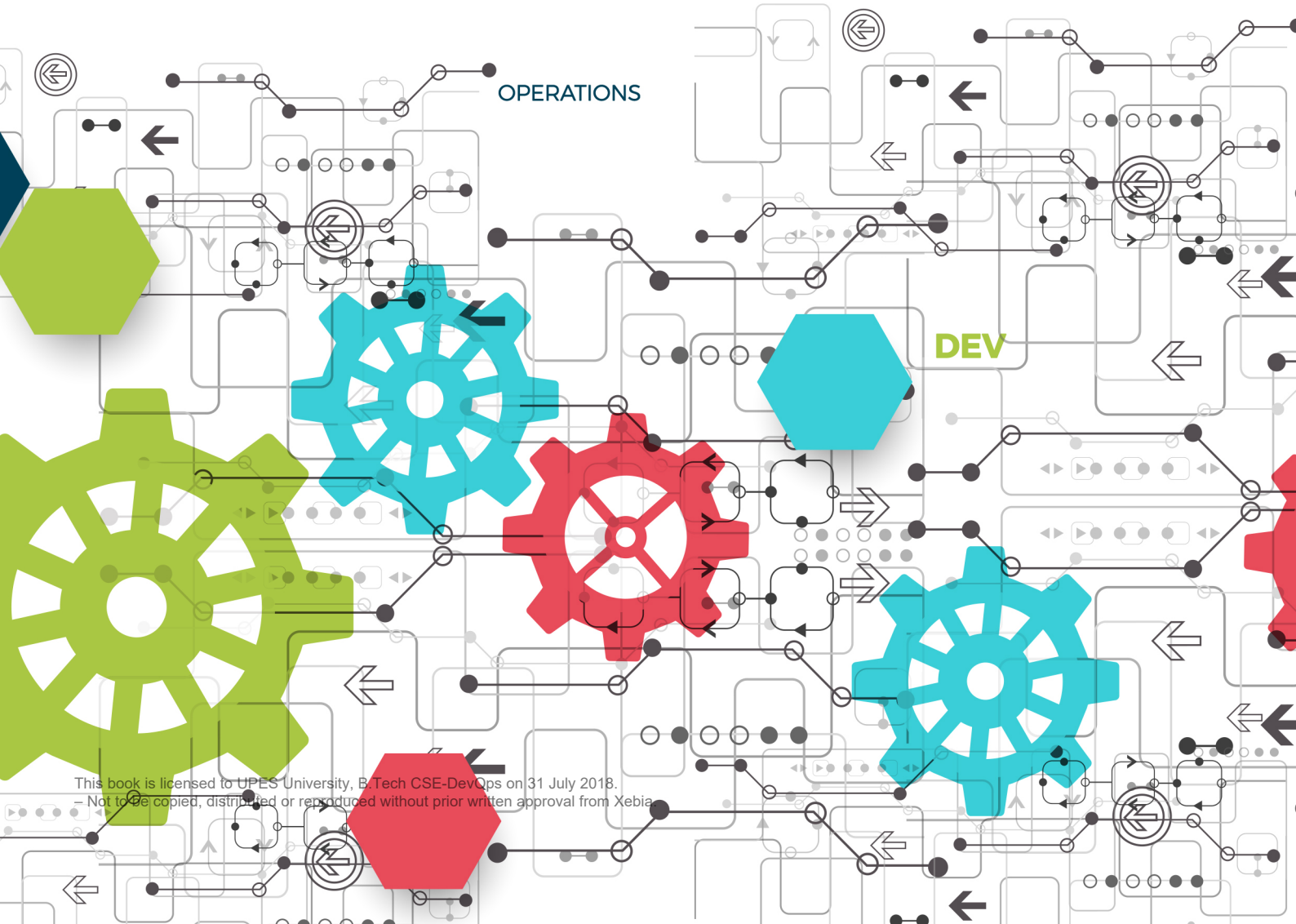
Xebia

B.Tech Computer Science and Engineering in DevOps

DEVOPS OVERVIEW

Semester **01** | **Student Handbook**

Release 1.0.0



Copyright & Disclaimer

B. TECH CSE with Specialization in DevOps

Version 1.0.0

Acknowledgements

We would like to sincerely thank the experts who have contributed to and shaped B. TECH CSE with Specialization in DevOps. Version 1.0.0

SME

Rajagopalan Varadan
A tech enthusiast who loves learning and working with cutting-edge technologies like DevOps, Big Data, Data science, Machine Learning, AWS & Open stack

Course Reviewers.

Aditya Kalia | Xebia
Maneet Kaur | Xebia
Sandeep Singh Rawat | Xebia
Abhishek Srivastava | Xebia
Rohit Sharma | Xebia

Review Board Members.

Anand Sahay | Xebia

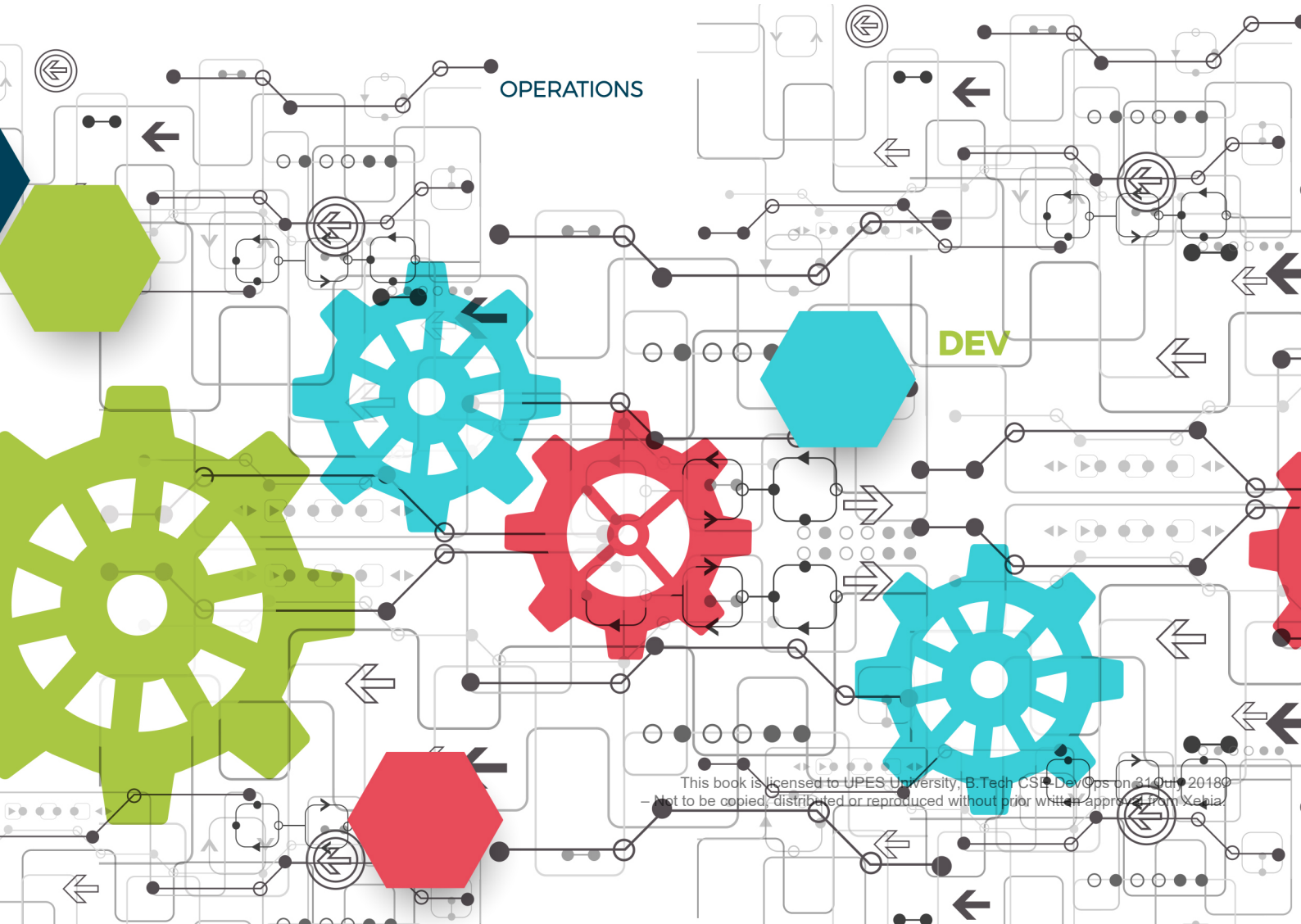
Xebia Group consists of seven specialized, interlinked companies: Xebia, Xebia Academy, XebiaLabs, StackState, GoDataDriven, Xpirit and Binx.io. With offices in Amsterdam and Hilversum (Netherlands), Paris, Delhi, Bangalore and Boston, we employ over 700 people worldwide. Our solutions address digital strategy; agile transformations; DevOps and continuous delivery; big data and data science; cloud infrastructures; agile software development; quality and test automation; and agile software security.

ODW is dedicated to provide innovative and creative solutions that contribute in growth of emerging technologies. As a learning experience provider, ODW strengths include providing unique, up to date content by combining industry best practices with leading edge technology. ODW delivers high quality solutions and services which focus on digital learning transformation.

B.Tech Computer Science and Engineering in DevOps

DEVOPS OVERVIEW

MODULE 2
Rise of Agile Methodologies



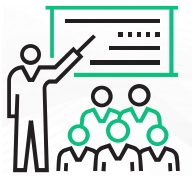
Contents

MODULE 2

Rise of Agile Methodologies

You will learn about DevOps Overview - Rise of Agile Methodologies in this module. Let's get started.

Module Learning Objectives



At the end of the Module you would be able to learn the following

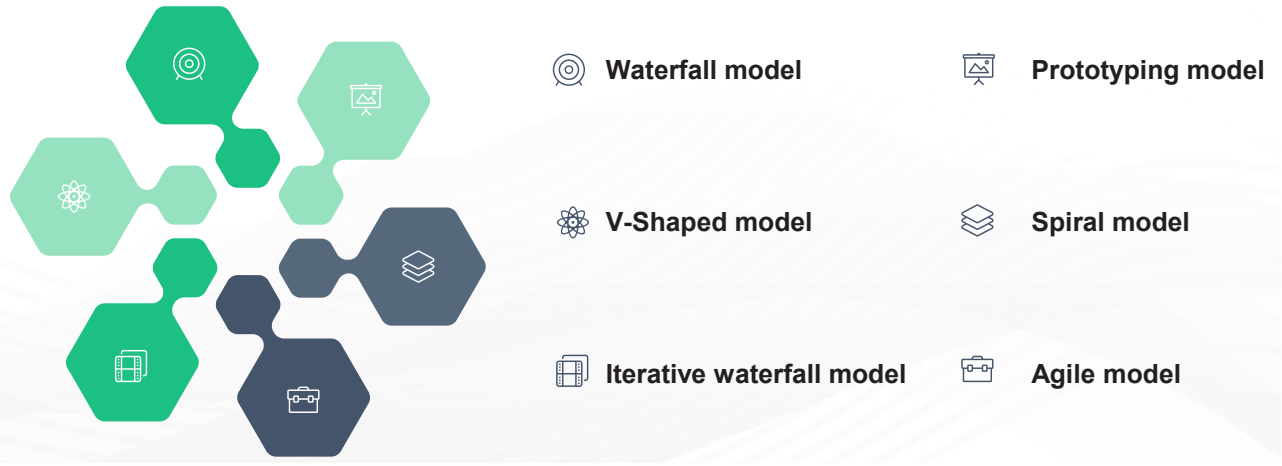
1. History of the rise of Agile.
2. Agile Manifesto - Values and Principles.
3. Comparison between Agile and the traditional Waterfall method of software development.
4. How a software is developed using Agile methodologies.
   a) The phases involved in the development cycle
   b) The four values of the Agile Manifesto

Module Topics

The following topics that will be covered in the module:

1. Agile movement in 2000.
2. Agile vs Waterfall Method.
3. Iterative Agile Software Development.
4. Individual and team interactions over processes and tools.
5. Working software over comprehensive documentation.
6. Customer collaboration over contract negotiation.
7. Responding to change over following a plan.

A Recap of Software Development Lifecycle Models



Software development life cycle (SDLC) is a series of phases right from business understanding and requirements gathering to software development till delivery of the final product. There are several approaches to software development.

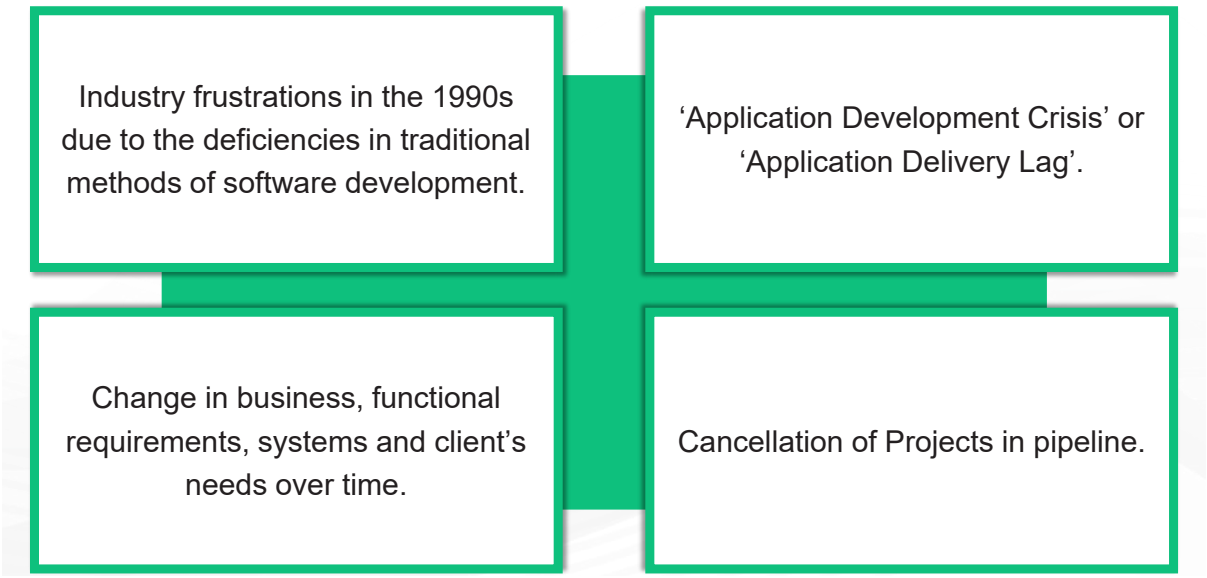
In the previous module we learnt in detail about the different software development methodologies.

- Waterfall model
- V-shaped model
- Iterative Waterfall model
- Prototyping model
- Spiral model
- Agile model

Though Traditional methods of software development are mature and usable, they created frustrations in the industry. Each of the events in the life cycle represents a distinct stage and completion of one stage is essential for the next stage to commence. Before the design phase begins, requirements must be reviewed and approved by the customer. Also, all deliverables are based on documented requirements and there exists a possibility of customer not seeing the final product until it's almost finished.

To overcome the shortfalls in traditional software development, industry experts identified and designed a set of models that are flexible, fast, lean, responsive and consistent. Software development methodologies that meet out the above mentioned criteria are collectively termed as agile methodologies. These methodologies emphasize rapid delivery of application as complete functional components. In the upcoming sections, we will see in detail about the agile movement and the rise of agile methodologies.

1. What Triggered the Rise of Agile?

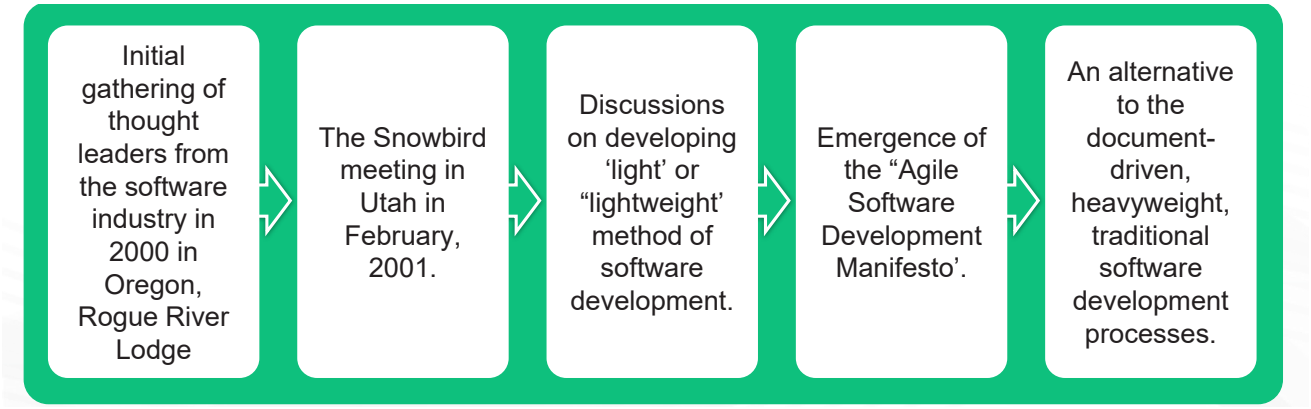


Around the 1990s, the software development industry was in a great crisis. Due to the deficiencies in the traditional software development approaches, there was an enormous lag in time between the functional requirements requested by the customers and the delivery of technology. At that time, it was estimated that the time period between business need validation and the delivery of software/ application was around three years. Some businesses faced a lag of more than three years.

Three years is a more than sufficient time period for businesses, systems and requirements to change. This resulted in the cancellation of projects mid-way. Completed projects failed to make sense, because of the change in requirements over time.

Industry experts were forced to develop alternative methods of software development and delivery, and this laid the foundation for the birth of Agile.

1.1 Birth of Agile



The initial meeting of seventeen thought leaders including Jon Kern, Kent Beck, Ward Cunningham, Arie van Bennekum, and Alistair Cockburn that happened in Oregon, voiced out the need for a 'light' or

'lightweight' development methodology. But nothing substantial happened at the end of this meeting. In 2000, a lot of articles were written which emphasized the importance of 'lightweight' methodologies. 'Light' was one term, which was used during that period, though no term was finalized,

Again in September 2000, Bob Martin from Object Mentor in Chicago invited the other thought leaders for a meeting that was scheduled to happen in January/February 2001.

Agile was born out of the historic 2001 Snowbird Lodge meeting, where advocates of Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development and Pragmatic Programming gathered to discuss the need for an alternative to the document-driven, heavyweight, traditional software development processes.

This group of thought leaders named themselves as the 'Agile Alliance' and agreed on the 'Manifesto for Agile Software Development.' It is in this Snowbird meeting, the Agile Manifesto and the Twelve Principles were formally written.

1.2 The Agile Manifesto

Developed by a group of fourteen thought leaders and practitioners from the software industry.

The essence of their experience that tells us what approaches work for software development and what does not.

Set of best practices involved in the development and delivery of high-quality software in shorter time periods.

THE AGILE MANIFESTO

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

→ Individuals and Interactions
Over Processes and Tools

→ Working Software
Over Comprehensive Documentation

→ Customer Collaboration
Over Contract Negotiation

→ Responding to Change
Over Following a Plan

The thought leaders who authored the 'Agile Manifesto'-

1. Mike Beedle
2. Arie van Bennekum
3. Alistair Cockburn
4. Ward Cunningham
5. Martin Fowler
6. Jim Highsmith
7. Andrew Hunt
8. Ron Jeffries
9. Jon Kern
10. Brian Marick
11. Robert C. Martin
12. Ken Schwaber
13. Jeff Sutherland
14. Dave Thomas

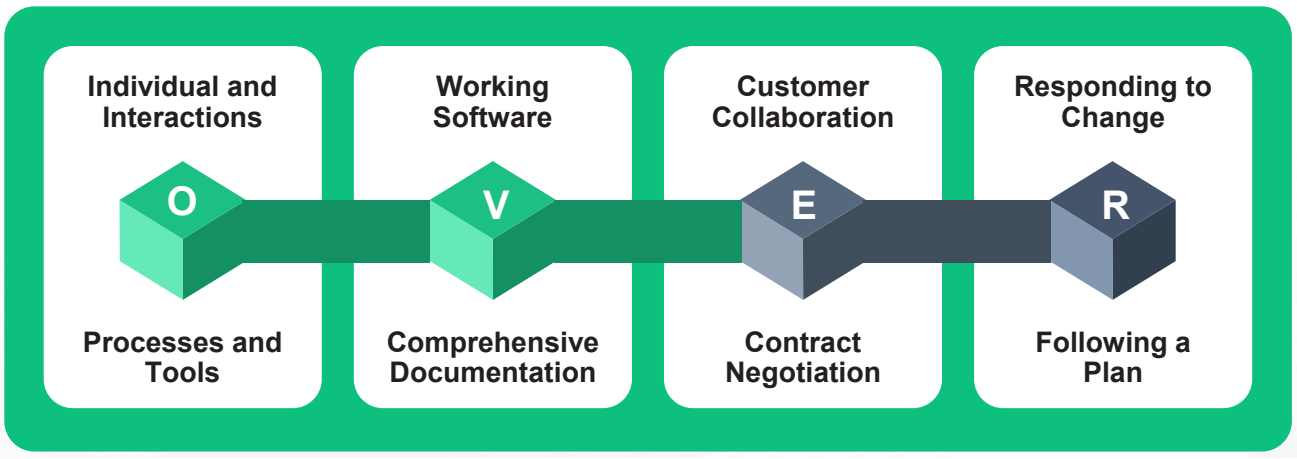
The manifesto reads:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

"Individuals and interactions over processes and tools.
Working software over comprehensive documentation.
Customer collaboration over contract negotiation.
Responding to change over following a plan."

"That is, while there is value in the items on the right, we value the items on the left more."

1.2.1 Four values of the Agile Manifesto



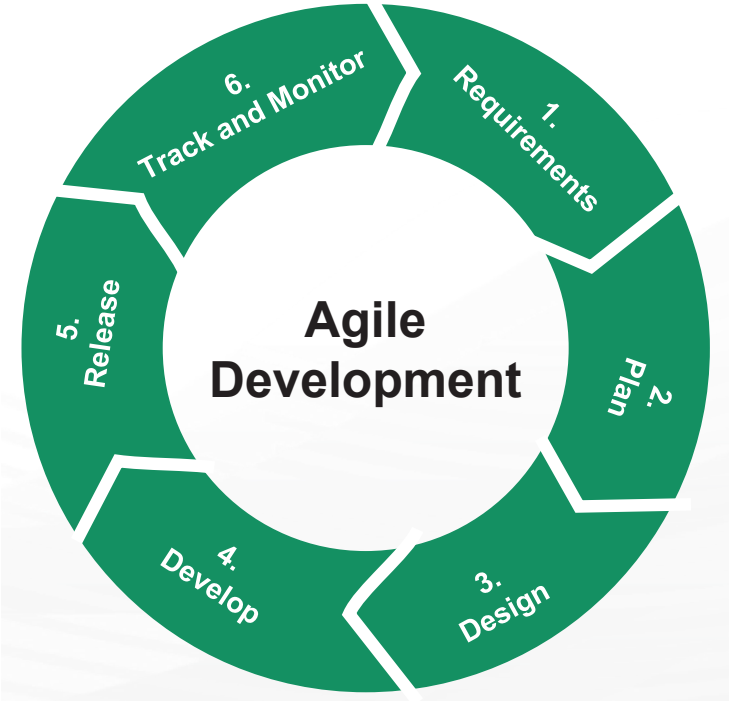
Each of these four values is described in a later section.

1.2.2 Twelve Principles of the Agile Manifesto

1	Our highest priority is to satisfy the customer through an early and continuous delivery of valuable software.
2	Welcome changing requirements, even late in the 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.
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 a 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.
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 behaviour accordingly.

Source: agilemanifesto.org

1.3 What is Agile Development?



- A time-bound, iterative approach to software development and delivery.
- Agile is about building and delivering software incrementally, instead of delivering it in one single shot at or near project end.

- Requirements and solutions evolve through collaboration between self-organized, cross-functional teams.
- Adaptive development process that fits changing business needs.
- In a nutshell, agile development refers to any product development process that is in lines with the Agile Manifesto.

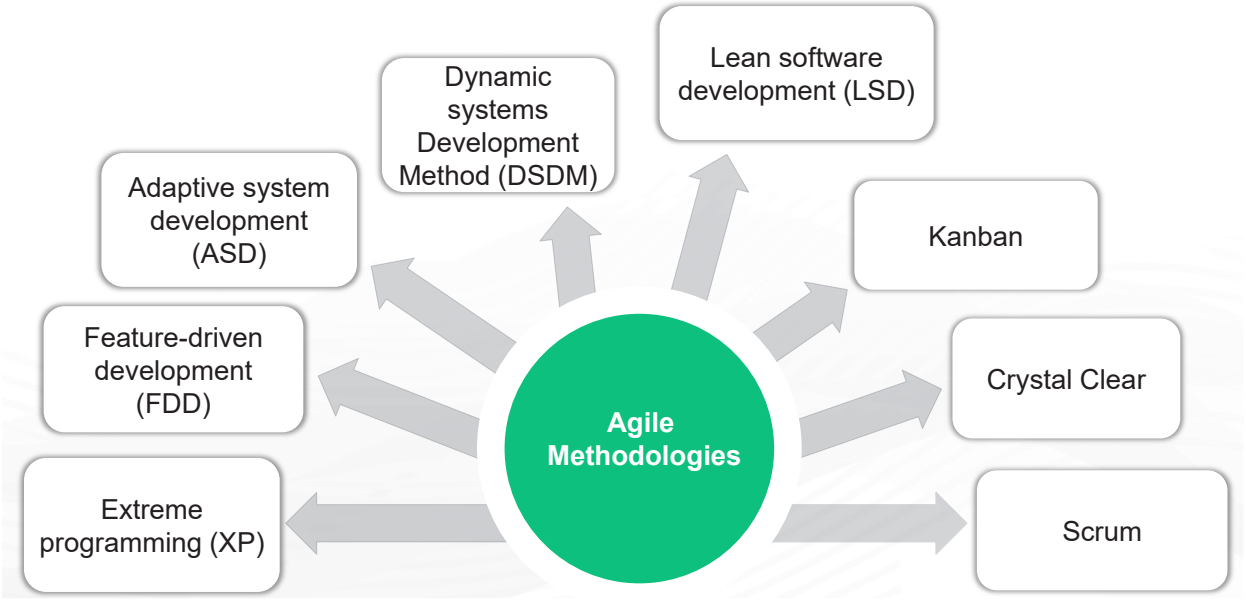
Major concepts of Agile development involve iterative development, frequent quality checks and feedbacks, self-organization of teams, accountability, adaptation to changes and aligning development approaches in lines with the customer's business requirements.

Agile breaks the product down into multiple functional units, based on user stories. These units are then prioritized for continuous delivery of software in short cycles called iterations.

Involves feedback loops to help teams to identify and develop solutions for the ever-changing business needs.

Agile development cycle is explained in detail in a later section.

1.4 Agile Methodologies



Extreme Programming (aka XP): Extreme programming is one of the most popular Agile methodologies widely adopted by many companies across the globe and has proven successful. This method puts a lot of emphasis on customer satisfaction. Improvement in quality and responsiveness according to changing customer needs, is the major intention behind the development of this methodology. The five essential ways by which this methodology improves a software project:

- Communication
- Simplicity

- Feedback
- Respect
- Courage

Developers can address change in customer requirements even later during the development cycle. The development team self-organizes around the problem to solve it as efficiently as possible. Unproductive steps in the product life cycle are cut down to reduce cost and frustration of the team.

Feature-driven Development (FDD): The five basic activities involved in FDD are:

- Develop overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature

This method blends industry best practices into one approach.

Adaptive System Development (ASD): This method requires the projects to be in a continuous state of adaptation. This method has a set of three repeating series:

- Speculate
- Collaborate
- Learn

Dynamic Systems Development Method (DSDM): Developed for addressing common failures of IT projects, like budget issues, missing deadlines, and lack of user involvement. There are eight principles in DSDM:

- Focus on the business need
- Deliver on time
- Collaborate
- Never compromise quality
- Build incrementally from firm foundations
- Develop iteratively
- Communicate continuously and clearly
- Demonstrate control

This methodology is useful in development and delivery of software and non-IT solutions.

Lean Software Development (LSD): In this methodology, lean manufacturing and lean IT principles are applied in software development. Seven principles that govern LSD are:

- Eliminate waste
- Amplify learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

Kanban: A Japanese term, that refers to 'a visual signal' or 'card'. A visual framework for implementing Agile workflow. Encourages the addition of small, but continuous changes to the system. Kanban works on the basis of the following principles:

- Visualize the workflow
- Limit work in progress
- Manage and enhance the flow
- Make policies explicit
- Continuously improve

Helpful in visualizing the build-up of work, roadblocks, reducing waste and maximizing value.

Crystal clear: Ideal for teams with 6 to 8 developers. This framework gives importance to people over processes and artifacts. Belongs to the Crystal family of methodologies. Principles include:

- Frequent delivery of usable code to users
- Reflective improvement
- Osmotic communication preferably by being co-located

Scrum: One of the most popular way to implement Agile. An open development framework, with a simple set of rules, responsibilities and meetings. Suggests dividing the projects into multiple short sprints, that usually last for one to two weeks. Team can thus deliver software in multiple iterations on a regular basis. Scrum emphasizes the following principles:

- Empirical feedback
- Team self-management
- Product increments within short iterations

What did you Grasp?

1. Which of the following is NOT true about with respect to the principles of Agile Manifesto?

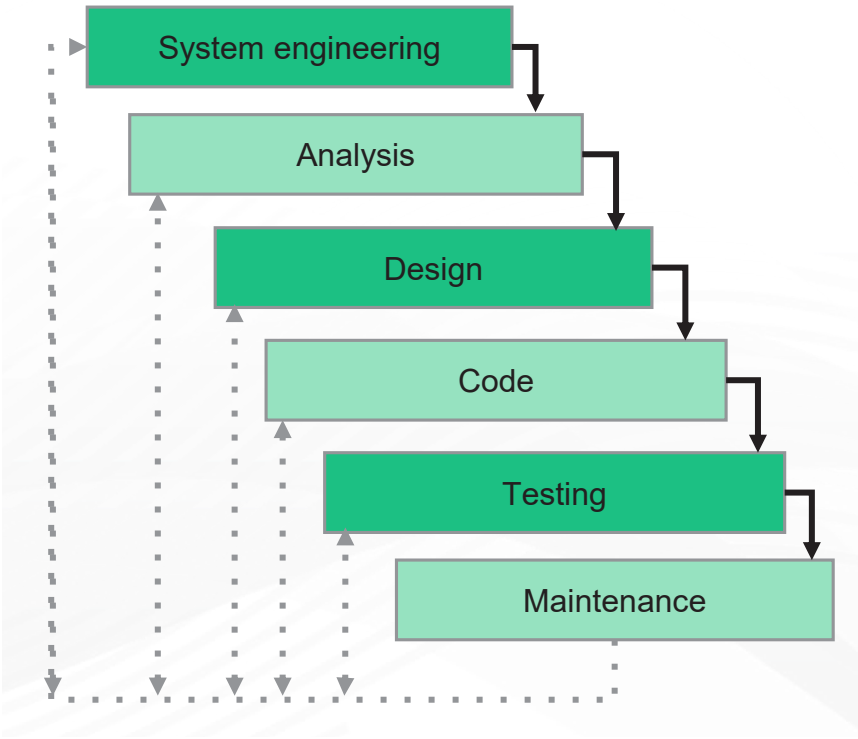
   A) Customer satisfaction by means of early and continuous delivery is the primary focus
   B) Completion of full-fledged technical documentation is considered one of the measures of progress
   C) Change in requirements to be accommodated even in later stages of development
   D) Agility is enhanced by attention to technical details and design

2. Which of the following Agile tools is helpful in visualizing the build-up of work?

   A) Scrum
   B) Lean Software Development
   C) Kanban
   D) Extreme Programming

2. The Waterfall Model - a recap

- A sequential model of software development, with a steady downward progression (one phase after the other).
- Each phase has defined set of features, tasks and deliverables
- One phase has to be completed before the next begins.
- Once completed, it's difficult to go back to the previous phase without scratching the project.
- Emphasizes the need for a complete documentation of requirements, before actual work starts.

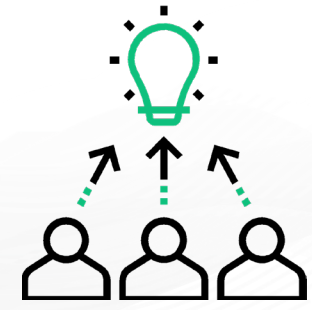


- Recall what you know about the Waterfall Model.
- Discuss and participate in the class discussion.

2.1 Phases in the Waterfall Model

Participants would have studied each of these phases in details in Module 01.

2.2 Activity



- Participate in the group activity.
- Place the points on the paper.

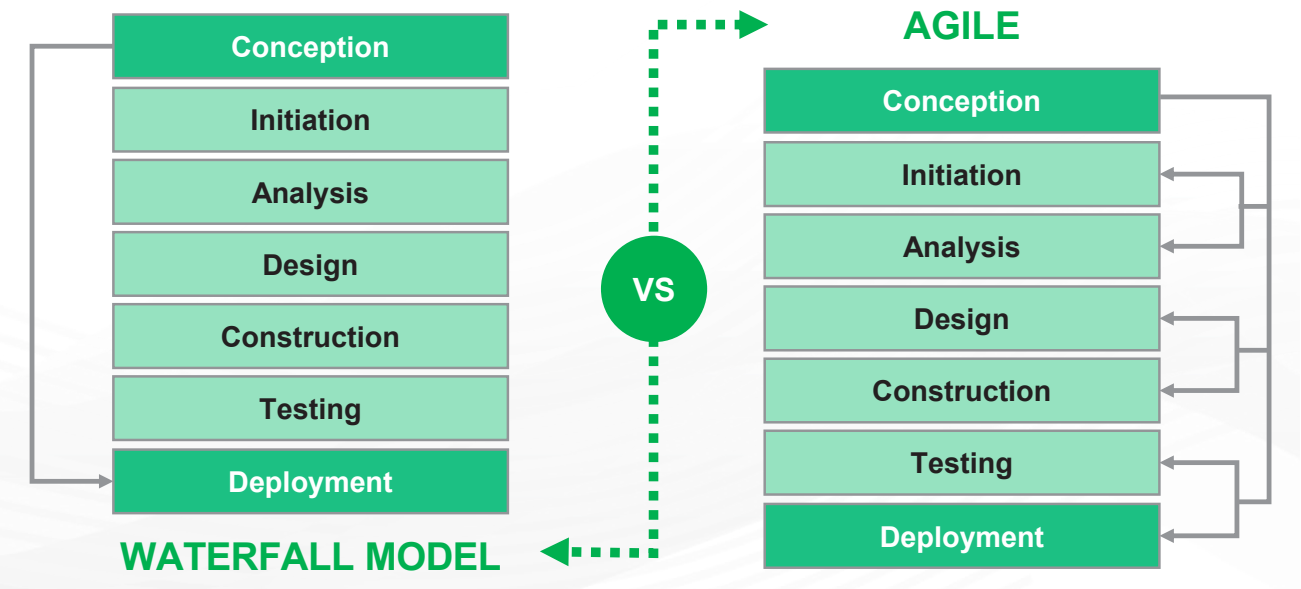
2.3 Comparing Agile and Waterfall

Agile	Waterfall
Agile follows an iterative and incremental approach, as weekly or monthly sprints.	Waterfall follows a sequential development approach.
The Software is developed and delivered in multiple iterations as small modules. There is always room for resolving errors and changes can be made as the current business requirement.	The whole project is delivered at the end, cannot handle any error or change in requirements.
Believes in developing working software over creating detailed documentation.	The Project will not commence until the complete documentation is ready.
Adapts to change in the scope of the project without causing much impact on timelines and budget.	Waterfall depends heavily on the initial requirements. Customers might not be able to demand change and if it happens it heavily impacts the budget and timelines.

Look at the table and understand the differences in the Agile and Waterfall method.

Agile	Waterfall
Project priorities are evaluated at the end of each sprint. Clients can add feedback then and there, product evolves in the desired shape.	Customers can share feedback only after the complete product is delivered, errors or enhancements can cause the project to be scratched and started from the beginning.
Testing is done at the end of every sprint. Any bug is identified early in the development cycle and fixed. Causes little or no impact on other code.	Testing is done only after the product is complete. Any bug that is created early, but discovered later can have an impact on rest of the code.
The Agile is used when rapid production is needed.	The Waterfall is used when product definition is more important than speed.
Though Agile follows iterative development approach, there might be a lack of the clear picture of what needs to be expected. The final product might be grossly different than what was intended initially.	The Waterfall relies on definitions and documentation, both the developers and customers have a clear idea of what will be built finally.

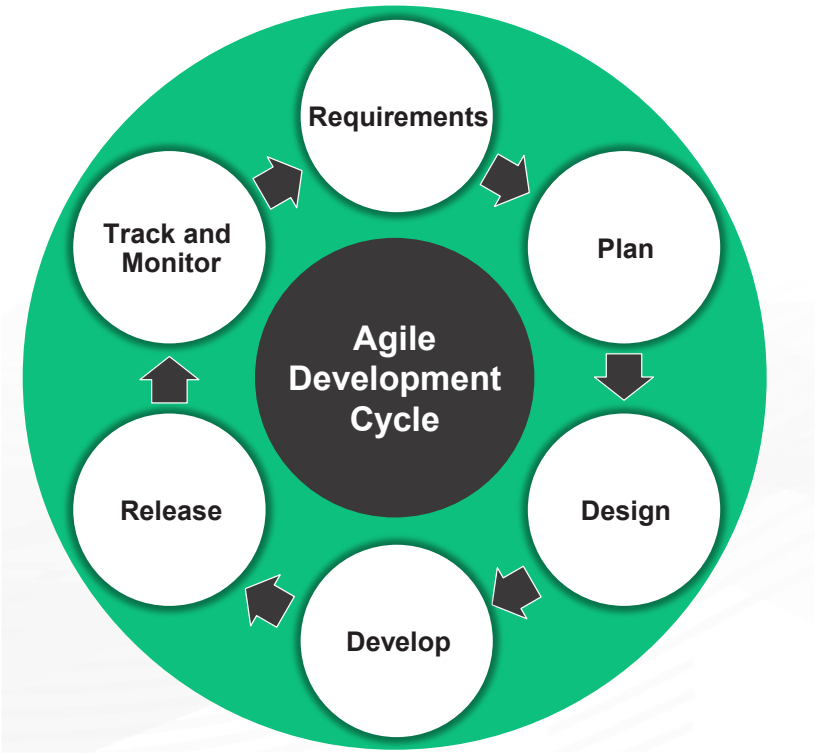
2.4 Comparing Agile and Waterfall



What did you Grasp?

1. Which of the following is TRUE with respect to waterfall development? Select all the correct options:
   - **A) Waterfall follows an incremental development approach**
   - **B) Waterfall requires the documentation to be completed before the commencement of development.**
   - **C) Waterfall adapts to the changes that happen during all the stages of development.**
   - **D) Testing and bug fixing is done at the end of project completion.**
   - **E) Customers can share frequent feedbacks during the development phases.**

3. The Agile Development Cycle



Phases of the agile development cycle can be explained as follows. Unlike traditional development methods which are sequential, any of the phases can happen in parallel in an agile development process.

Planning: During this phase, the development team collaborates and identifies the scope of the project. The main aim of this phase is to break the idea down into smaller feature sets/tasks. Features can be allocated to iterations, based on priority. Priorities can change from iteration to iteration.

Analysis of Requirements: During this phase, many rounds of interactions happen between the stakeholders involved in the project, like product managers, cross-functional team members and customers. The goal of this phase is to identify the business requirements, understand the end user and how they will use the product. At the end of this phase, the teams will have a relevant and detailed information of the requirements.

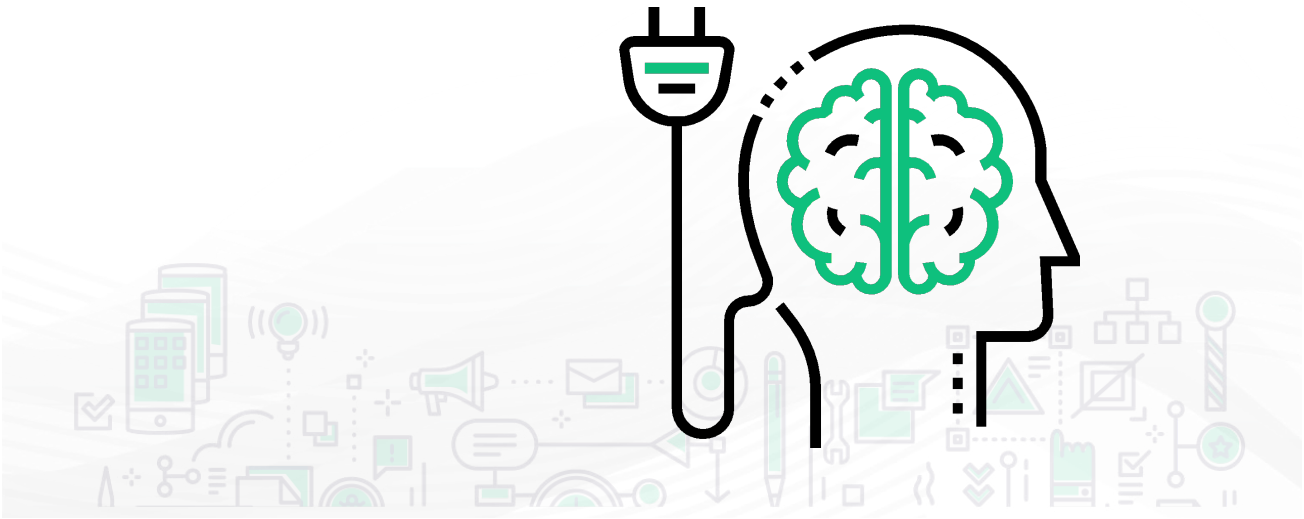
Design: In the design phase, documents on designing the systems and software is prepared, using the requirements gathered from the previous phase. During this phase, the team works on the usability part of the product, i.e., what the product or solution will look like and how users will interact with the system. A test plan is also drafted during this phase.

Implementation, coding & development: This phase involves development (coding) and testing of features and scheduling iterations for deployment. The team follows Iterative and Incremental Development approach for scheduling the iterations. The development phase has the first iteration being termed iteration 0 (zero), since no feature is delivered in this iteration. This is the most important and foundational phase of the development that involves finalization of contracts, preparation of environments and fund allocation.

Testing: Once development commences, the code is tested to find if it is matching the business needs of the customer. This phase also ensures that the product solves customer's requirements at each and every iteration. Different testing processes involved in this phase are unit testing, system testing and acceptance testing.

Deployment: This is the phase where the product is delivered for customer's use, after being tested. Even after this phase, customers may come up with the new set of problems that need to be addressed by the team.

What did you learn today?



What did you Grasp?

1. The Phases in Agile Development can happen in parallel.
   A) True
   B) False

2. When is the project broken down into multiple feature sets?
   A) Design
   B) Development
   C) Planning
   D) Requirement analysis

4. Agile Manifesto Value

4.1 Agile Manifesto Value 1: Individual and team interactions over processes and tools

INDIVIDUALS AND INTERACTIONS

- Communication- the key to the success of a project
- People build Products
- Have the focus on people and the source of energy
- Self-organization of cross-functional teams - to identify scope, negotiate, accept, define, collaborate, share, and solve problems
- Individuals need to be motivated
- Interactions need to be fostered among team members, customers and other stakeholders

Prioritizing individuals and interactions over processes and tools is the first value of the Agile manifesto. People drive the development process since they are the ones who respond to changing business needs and develop processes and tools in response to change.

If a team prioritizes processes, technology or tools then the individuals in the team become less responsive and meeting the customer requirements become a difficult task.

Communication between the team members, customers and other stakeholders is critical for understanding the business requirements and delivering value. If individuals are valued over processes and tools, the communication becomes fluid and the interactions happen, based on the change in business requirements. If processes are valued more, the interactions become scheduled and less adaptive to changes. There is a possibility of the customer's requirements losing importance because of the stringent schedules.

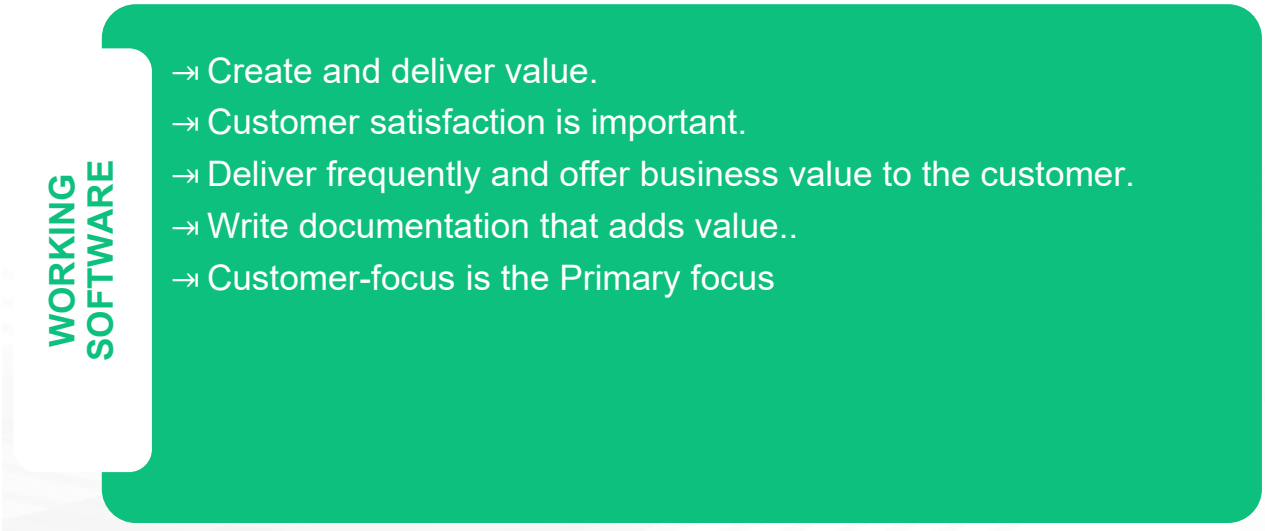
What did you Grasp?

1. The communication between teams will be fluid only if processes are valued over individuals.

   A) True

   B) False

4.2 Agile Manifesto Value 2: Working software over comprehensive documentation



The second value as per the agile manifesto is Working Software over Comprehensive Documentation.

Jim Highsmith, one of the authors of the Agile Manifesto and the primary developer of the “Adaptive Software Development” Agile Method, says, “We want to restore a balance. We embrace modelling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment.”

Writing pages and pages of documentation for developing the product, consumes an enormous amount of time, hence longer time periods between documentation and delivery. A lot of time, money and energy is spent in writing technical and functional specifications, customer’s business requirements, user interface specifications, design documents, documents on testing and much more. And the best part is that approvals are required for each and every document written by people, at multiple levels. This is the major cause of the delay in development and delivery of the actual product.

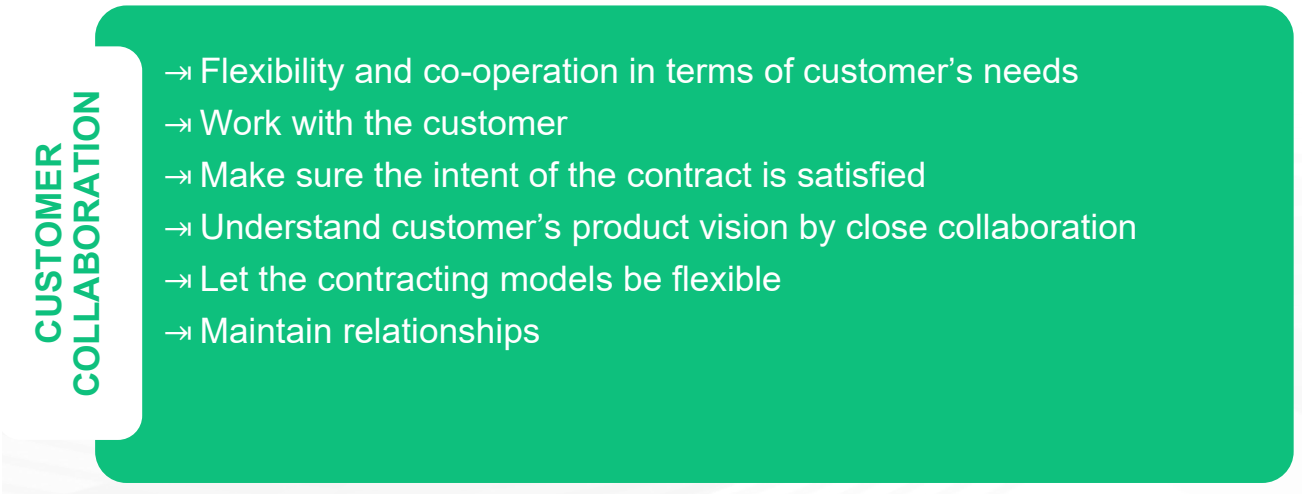
In agreement with Jim’s statement, Agile does not rule out the need for documentation. Agile emphasizes that the document is streamlined in a way that it gives the exact picture to the developer of what is exactly needed to build the software, without getting lost in the intricacies.

According to Agile methodologies, the requirements are documented as user stories, so the developer gets clarity on the business requirements. This eventually becomes a guide for the developer to start building the exact functionalities. Agile does give importance to documentation, but a working software is more critical from a customer’s viewpoint than documentation. Agile also emphasizes that delivering working pieces of the software at frequent intervals matters the most to the customer.

What did you Grasp?

1. Which of the following statements is NOT true?
   - **A) Agile completely rules out the need for technical documentation.**
   - **B) Agile emphasizes that documentation must add value to the developer and the user.**
   - **C) Delivering working software at frequent intervals is important to the customer**
   - **D) Agile considers user stories as the major source of reference for the developer.**

4.3 Agile Manifesto Value 3: Customer collaboration over contract negotiation



Customer Collaboration over Contract Negotiation is the third value described in the agile manifesto.

Negotiation enables the customer and the product manager to work out the details of product delivery. There is a room for re-negotiation as well. In traditional software development processes, customer engagement is more before the start of the development process and after the product is completely ready. There are two major disadvantages to this. Customer gives the complete specifications/ requirements in great detail, well before the development starts. There is a chance for confusions and multiple rounds of discussions happening even before the actual work starts, which causes the delay in the development. If the product is completely ready, and the customer starts changing the requirements, much effort will be involved in redoing things.

Agile manifesto emphasizes that the customer is engaged in and collaborates throughout the development process. Demo sessions should happen periodically. This enables feedback sharing at regular intervals and the product gets developed iteratively. The development team and the customer can make sure that the requirements are met at each and every point in time during the development process.

This value suggests the end - user being part of the development process, attending all the meetings and making sure changes are communicated then and there and ensuring the smooth delivery of the project.

What did you Grasp?

1. Select the correct statement.

   A) In Agile, customers do not engage with the team during the development phase.

   B) Agile does not promote customer collaboration, they might change the requirements frequently.

   C) Agile emphasizes the need for periodic demo sessions with the customer.

   D) There is no frequent feedback loop in Agile development.

4.4 Agile Manifesto Value 4: Responding to change over following a plan

RESPONDING TO CHANGE

- Change is the reality, the worst enemy of any plan
- Changes in customer's business needs - direct impact on developer's plans
- Adaptive planning for accepting change
- Change to be reflected in the product
- Show improvement

Responding to Change over Following a Plan is the fourth value as per the Agile Manifesto.

A Change was considered expensive and avoidable by the traditional software development methodologies. Planning was considered much more important than responding to changes. The intention was to have the detailed and comprehensive plans, where feature sets and functionalities are pre-defined. Since traditional methods follow a sequential order in the development and delivery of software, lot of unnecessary dependencies arise. High priority is given to each and every step of the development process.

On the other hand, Agile suggests shorter iterations. This enables change in priorities from iteration to iteration, according to the customer's requirements. There is room for the addition of new features in the next iteration, without having any dependency on the previous iteration. Agile views change as a means for improvement and believes that a change adds value to the product.

A process called Method Tailoring describes The Agile approach towards change. An Agile Information Systems Development Method defines Method Tailoring as a process or capability in which human agents determine a system development approach for a specific project situation through responsive changes in, and dynamic interplays between contexts, intentions, and method fragments."

With Agile, the process modifications that fit the team are allowed.

What did you Grasp?

1. Agile considers change as a driver for improvement.
   A) True
   B) False

In a nutshell, we learnt:

1. The birth of Agile to handle the disadvantages caused due to traditional software development methodologies.
2. The Four Values and Twelve principles of the Agile Manifesto.
3. Different types of Agile methodologies.
4. Comparison of Agile and Waterfall methodologies.
5. The Agile Development Cycle and Phases involved.
6. Explanation of the four values of the Agile Manifesto.

Note down the salient points of what you have covered here.

Release Notes

B. TECH CSE with Specialization in DevOps

Semester One -Year 01

Release Components.
Facilitator Guide, Facilitator Course Presentations, Student Guide and Mock exams.

Current Release Version.
1.0.0

Current Release Date.
2 July 2018

Course Description.
Xebia, has been recognized as a leader in DevOps by Gartner and Forrester and this course is created by Xebia to equip students with set of practices, methodologies and tools that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.

The language used in this course is US English. Our sources of reference for grammar, syntax, and mechanics are from The Chicago Manual of Style, The American Heritage Dictionary, and the Microsoft Manual of Style for Technical Publications.

Bugs reported	Not applicable for version 1.0.0
Next planned release	Version 2.0.0 Feb 2019