
Introduction to Agile
Lesson 1: Agile Process Framework
Capgemini

Lesson Objectives

- History of Traditional Software Development Model
- Software Development Model and SDLC
- "Waterfall Model" – An Overview
- Waterfall or Sequential Based Development Model
- "Real Life" – Waterfall Model
- "Waterfall Model" – Advantages
- "Waterfall Model" – Disadvantages
- Agile Software Development – Definition
- Agile Development Model
- Graphical Illustration of Agile Development Model
- Why use Agile?
- Agile Manifesto and Principles
- 12 Principles of Agile Methods

Lesson Objectives

- Agile Values
- What is NOT an Agile software development?
- Foundation of an Agile software development Method
- Common Characteristics of Agile Methods
- Agile Methods and Practices
- When to use Agile Model?
- Advantages of Agile Model
- Disadvantages of Agile Model
- Difference between Agile and Waterfall Model
- Agile – Myths and Reality
- Agile Market Insight

1.1: Overview of Traditional Software Development Model

History of Traditional Software Development Model

- Traditional software development methodologies are often called heavyweight methodologies as they are based on a sequential series of steps that has to be defined and documented in detail
- These methodologies are based on progressive series of steps like requirements definition, design and architectural planning, development and testing
- The traditional software development models depends upon a set of predefined processes
- The success of a project which is build with traditional software development model depends upon how well the requirements are stated before the project begins
- In this approach implementing changes during development life cycle is somewhat critical
- However, this approach also poses benefits like easy estimation of cost of the project, scheduling and allocation of resources

History of Traditional Software Development Model

The traditional software development models like Waterfall Model, V-Model and RUP are classified into the heavyweight methodologies. These methodologies are based on progressive series of steps like requirements definition, design and architectural planning, development and testing. Traditional software development methodologies require defining and documenting a stable set of requirements at the beginning of a project.
The traditional software development model comprises of four basic phases:
The first phase is to set up the system requirements for the project and defining the amount of time it will take to implement the various phases of development life cycle.
Once the requirements are finalized, the next step moves into the design and architectural planning phase. In this phase the technical infrastructure is created in the form of diagrams or models.
Once the team is satisfied with the architectural and design plan, the project moves into the development phase where code is produced until the specific goals are reached. Development is often broken down into smaller tasks that are distributed among various teams based on skill.
The testing phase often overlaps with the development phase to ensure issues are addressed early on. Once the project nears completion and the developers are close to meeting the project requirements, the customer will become part of the testing and feedback cycle and the project was delivered after the customer satisfy with it.
The success of a project which is build with traditional software development model depends upon how well the requirements are stated before the project begins. In this approach implementing changes during development life cycle is somewhat critical. However, this approach also poses benefits like easy estimation of cost of the project, scheduling and allocation of resources.

1.1: Overview of Traditional Software Development Model

Software Development Model and SDLC

- Software development models are various processes or methodologies used to develop the product
- Software developments models help improve the software quality as well as the development process in general
- There exists various software development models and each one of them fulfill certain objectives of software development
- Software Development Life Cycle (SDLC) is an environment that describes activities performed in each stage of the software development process
- The SDLC contains detailed plan which basically describes how the development and maintenance of specific software is conducted
- Most people involved with software development are very much familiar with the traditional software development methods like:
  - Waterfall or the sequential method
  - V-model

Software Development Model and SDLC

A software application or an information system is designed to perform a particular set of tasks. Often, this set of tasks that the system will perform provides well-defined results, which involve complex computation and processing. Therefore it's a very important and tedious job to administer the entire development process to ensure that the end product comprises of high degree of integrity and robustness, as well as user acceptance.

Software Development Life Cycle (SDLC) is a process of building or maintaining software systems. The SDLC process primarily takes care of various pre-development phases like requirement gathering, requirement analysis, design and architecture as well as post development activities like testing and validation. It also consists of the models and methodologies that development teams use to develop the software systems, which the methodologies form the framework for planning and controlling the entire development process.

Currently, there are two SDLC methodologies which are utilized by most system developers, namely the traditional development and agile development.

Most people involved with software development are very much familiar with the traditional software development methods like:

1. Waterfall or the sequential method
2. V-model

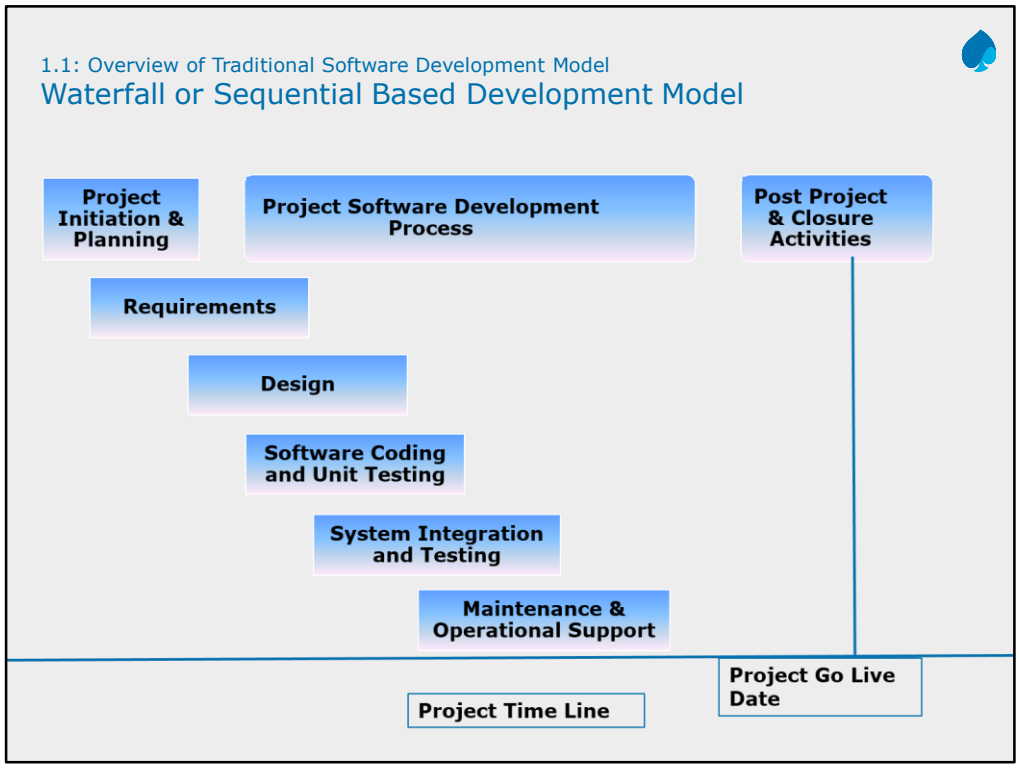
1.1: Overview of Traditional Software Development Model

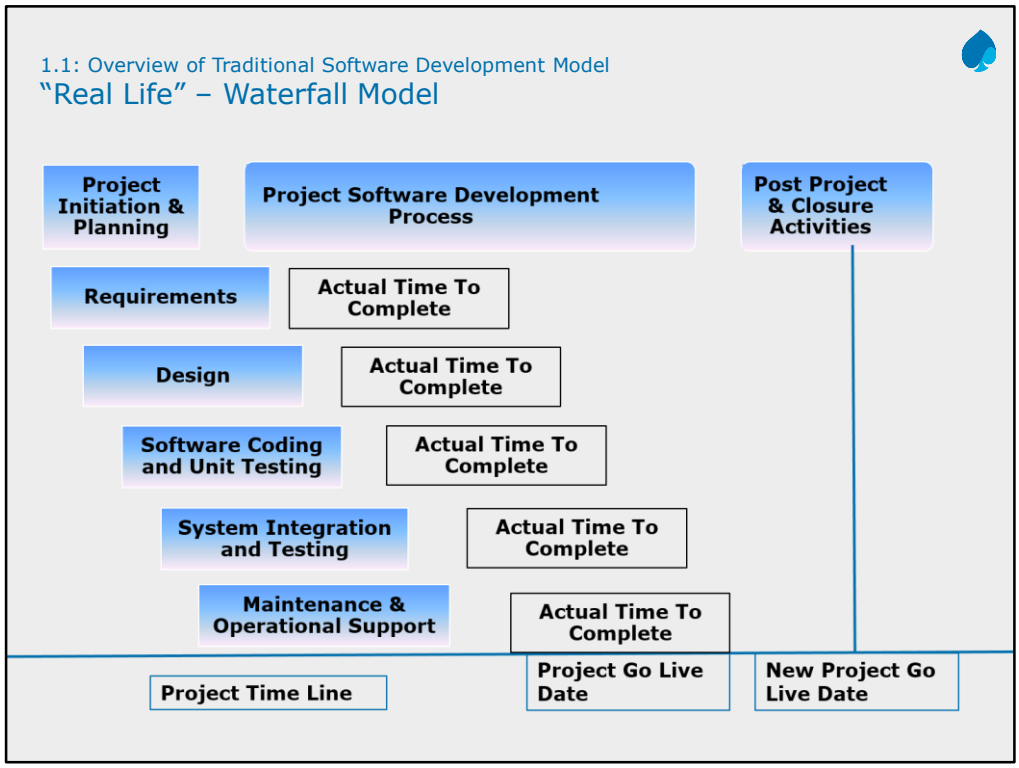
"Waterfall Model" – An Overview

- The classic waterfall model was introduced in the 1970s by Win Royce
- The Waterfall Model was the first Process Model to be introduced
- It is also referred to as a linear-sequential life cycle model
- The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards like a waterfall through the phases of SDLC
- Every stage has to be completed separately at the right time so you can not jump stages
- Documentation is produced at every stage of a waterfall model to allow people to understand what has been done
- Testing is done at every stage
- The waterfall approach assumes that requirements are stable and frozen across the project plan
- However, this is usually not true in case of large projects where requirements may evolve across the development process

Waterfall or Sequential Based Development Model

The waterfall model was very simple and logical in the way it approached software development and for these reasons alone it is still in use today. It takes very little time for anybody new to a development team or project to understand a sequential based process.
While it may be logical, one of its biggest problems was ensuring that the time spent in each phase didn't overrun into the next one resulting in a series of compounding delays. Unfortunately, this was the case more times than not and ultimately the testing phase got "squeezed" against the deadline, which became an immovable date.
Or if the project timelines are fixed then either parts of the original scope and functionality may not be delivered or even worse the functionality is delivered, but there is no time for sufficient testing and the quality of the product suffers.


1.1: Overview of Traditional Software Development Model
Waterfall or Sequential Based Development Model
Project Initiation & Planning
Project Software Development Process
Post Project & Closure Activities
Requirements
Design
Software Coding and Unit Testing
System Integration and Testing
Maintenance & Operational Support
Project Go Live Date
Project Time Line


1.1: Overview of Traditional Software Development Model
"Real Life" – Waterfall Model
Project Initiation & Planning
Project Software Development Process
Post Project & Closure Activities
Requirements
Actual Time To Complete
Design
Actual Time To Complete
Software Coding and Unit Testing
Actual Time To Complete
System Integration and Testing
Actual Time To Complete
Maintenance & Operational Support
Actual Time To Complete
Project Time Line
Project Go Live Date
New Project Go Live Date

1.1: Overview of Traditional Software Development Model

"Waterfall Model" – Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model, each phase has specific deliverables and a review process
- The project requires the fulfilment of one phase, before proceeding to next
- Works well for smaller projects where requirements are very well understood
- Various stages of the software development can be clearly defined in waterfall model
- Well understood milestones
- A schedule of activities can be created with deadlines for each stage of development
- Product development progresses from vision, through design, implementation, testing, and ends up at operation and maintenance
- Each phase of development proceeds in strict order

"Waterfall Model" – Advantages

Waterfall model is simple to implement and also the amount of resources required for it are minimal. In this model, output is generated after each stage (as seen before), therefore it has high visibility. The client and project manager gets a feel that there is considerable progress. Here it is important to note that in any project psychological factors also play an important role.
Project management, both at internal level and client's level, is easy again because of visible outputs after each phase. Deadlines can be set for the completion of each phase and evaluation can be done from time to time, to check if project is going as per milestones.
This methodology is significantly better than the haphazard approach to develop software. It provides a template into which methods of analysis, design, coding, testing and maintenance can be placed.
This methodology is preferred in projects where quality is more important as compared to schedule or cost.

1.1: Overview of Traditional Software Development Model

"Waterfall Model" – Disadvantages

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage
- The customer can experience the working model of the product only at the end
- Not suitable for complex & large projects
- Only a certain number of team members will be qualified for each phase, which can lead at times to some team members being inactive
- It is difficult to follow the sequential flow in software development process

"Waterfall Model" – Disadvantages

The Waterfall model exhibits poor flexibility. The majority of software is written as part of a contract with a client, and clients are notorious for changing their stated requirements. Thus the software project must be adaptable, and spending considerable effort in design and implementation based on the idea that requirements will never change is neither adaptable nor realistic in these cases. The waterfall model however is argued by many to be a bad idea in practice, mainly because of their belief that it is impossible to get one phase of a software product's lifecycle "perfected" before moving on to the next phases and learning from them. A typical problem is when requirements change midway through, resulting in a lot of time and effort wastage due to "Big Design Up Front".
Constant testing from the design, implementation and verification phases is required to validate the phases preceding them. Users of the waterfall model may argue that if designers follow a disciplined process and do not make mistakes that there is no need to constantly validate the preceding phases.

1.2: Agile Process Framework

Agile Software Development – Definition

- Wikipedia defines Agile Software Development as :

> Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle.

1.2: Agile Process Framework

Agile Software Development – Definition

- The web site, SearchSoftwareQuality.com contains the following definition:

> In software application development, agile software development (ASD) is a methodology for the creative process that anticipates the need for flexibility and applies a level of pragmatism into the delivery of the finished product. Agile software development focuses on keeping code simple, testing often, and delivering functional bits of the application as soon as they're ready. The goal of ASD is to build upon small client-approved parts as the project progresses, as opposed to delivering one large application at the end of the project.

1.2: Agile Process Framework

Agile Development Model

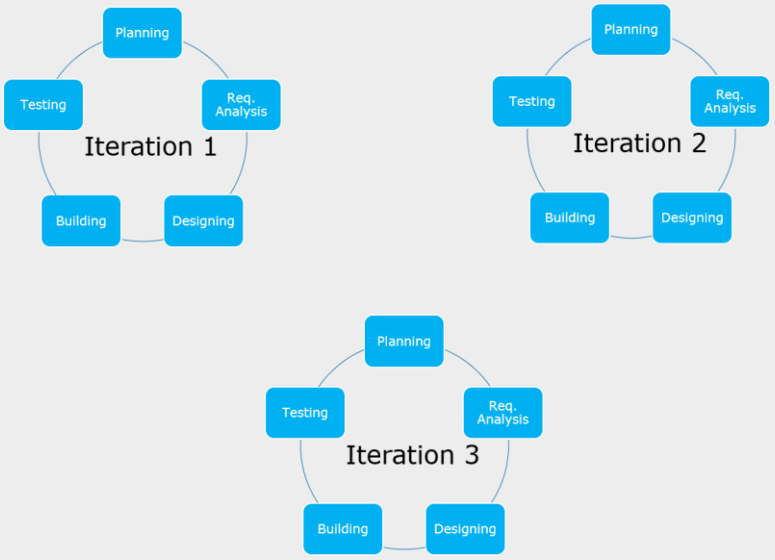
- Agile development model is a an amalgamation of iterative and incremental process models focusing more on process adaptability and customer satisfaction by rapid delivery of functional software product
- Agile development model breaks the software into small incremental builds
- These builds are provided in iterations
- Each iterations lasts from about one to three weeks
- Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing
- At the end of the iteration a working product is displayed to the customer and important stakeholders

Agile Development Model

Agile Development Model works on the basis of modern development approach which strongly believes that every project and every project needs are different. To attain the project success one needs to make an effort to tailored the existing methodologies to best suit the changing project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release. Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.
Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability. The most popular agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as agile methodologies, after the Agile Manifesto was published in 2001.

1.2: Agile Process Framework

Graphical Illustration of Agile Development Model



1.2: Agile Process Framework

Why use Agile?

- Improved return on investment (RIO)
- Early detection and cancellation of failing products
- Higher quality software
- Improved control of a project
- Reduced dependence on individuals and increased flexibility

Why use Agile?

Improved return on investment: This is the fundamental reason to use agile methods, and it's achieved in a number of different ways. In an agile project, the initial requirements are the baseline for ROI. If the project runs to completion with no changes, then the business will get the projected returns. However, by providing frequent opportunities for customer feedback, agile methods let customers steer the project incrementally, taking advantage of new insights or changed circumstances to build a better system and improve the ROI. By delivering working software early and often, agile projects also present opportunities for early deployment and provide earlier return on smaller initial investments.

Early cancellation of failing projects: It's common to observe projects to be on track for the first 80% or more of the schedule, only to be delayed for many months. In this situation sponsors face a difficult choice. Do they abandon the project after spending 80% of the budget or continue to fund the project in the hope of getting some return on the investment?

Higher quality: Of the four fundamental variables you can use to control a software development project, cost, time, scope and quality, most agile methods explicitly use scope as their control variable. All agile methods emphasize the production of high quality software, and extreme programming in particular adds a number of practices to support this objective.

Agile Manifesto and Principles

The agile manifesto is the result of a meeting at the Snowbird ski resort in Utah in 2001. Prior to that date, the individual agile processes were referred to as lightweight.
The release of the manifesto immediately gave the industry a tangible definition of agile and ground rules for adding new ideas in the future. The manifesto provides clear direction and is used to discuss and compare agile methodologies. The manifesto provides one common roof for all agilists, whatever their favorite agile methodology might be.

Here are the core values of the manifesto:

1. **Individuals and interaction take precedence over processes and tools.**
2. **Working software takes precedence over comprehensive documentation.**
3. **Customer collaboration takes precedence over contract negotiation.**
4. **Responding to change takes precedence over following a plan.**

Please note that the left side of each statement is valued more than the right side. What is important and often misunderstood is that the manifesto does not recommend neglecting the values of the right side - for example, project documentation. It simply means that the values on the left are valued more highly. Every agile project team has to find the right balance as a team, but also they must find balance within the organization.

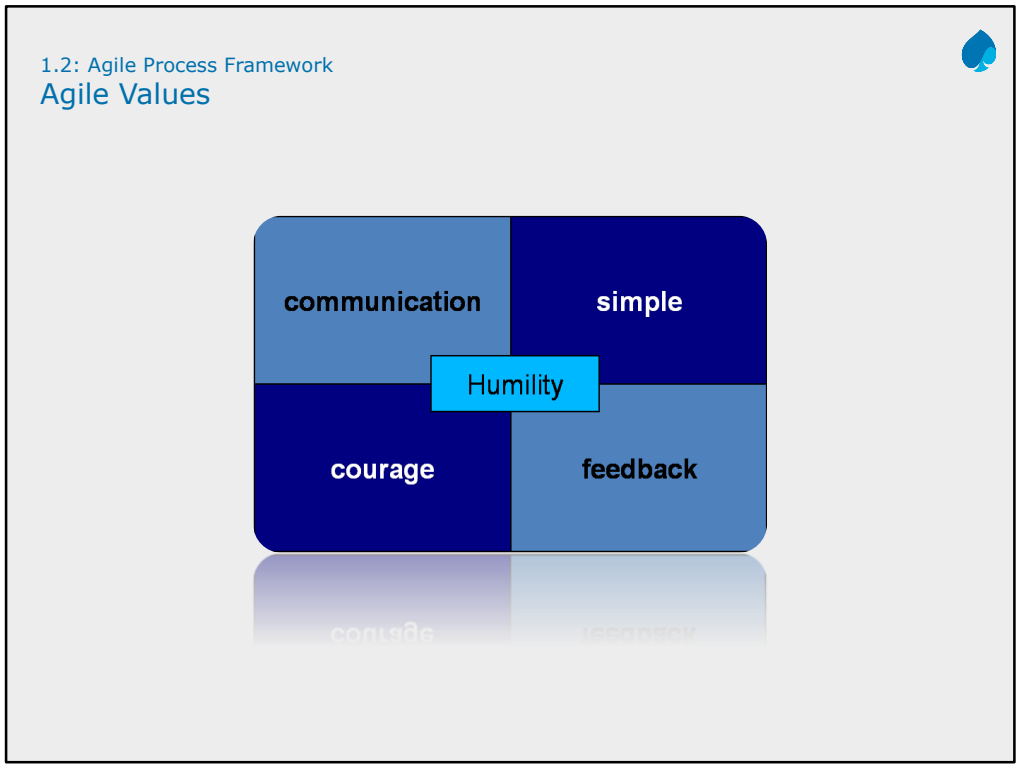
1.2: Agile Process Framework

12 Principles of Agile Methods

- According to Kent Beck, the Agile Manifesto is based on twelve principles:
  - Customer satisfaction by rapid delivery of useful software
  - Welcome changing requirements, even late in development
  - Working software is delivered frequently (weeks rather than months)
  - Working software is the principal measure of progress
  - Sustainable development, able to maintain a constant pace
  - Close, daily cooperation between business people and developers
  - Face-to-face conversation is the best form of communication (co-location)
  - Projects are built around motivated individuals, who should be trusted
  - Continuous attention to technical excellence and good design
  - Simplicity—the art of maximizing the amount of work not done - is essential
  - Self-organizing teams
  - Regular adaptation to changing circumstances

The Principles of Agile Methods – www.agilemanifesto.org

1. Our highest priority is to satisfy the customer through the 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.
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.
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.



The five values of Agile Modelling (AM) are:

1. Communication: Models promote communication between your team and your project stakeholders as well as between developers on your team.

2. Simplicity: It's important that developers understand that models are critical for simplifying both software and the software process, it's much easier to explore an idea, and improve upon it as your understanding increases, by drawing a diagram or two instead of writing tens or even hundreds of lines of code.

3. Feedback: Kent Beck says it best in Extreme Programming Explained "Optimism is an occupational hazard of programming, feedback is the treatment." By communicating your ideas through diagrams, you quickly gain feedback, enabling you to act on that advice.

4. Courage: Courage is important because you need to make important decisions and be able to change direction by either discarding or refactoring your work when some of your decisions prove inadequate.

5. Humility - The best developers have the humility to recognize that they don't know everything, that their fellow developers, their customers, and in fact all project stakeholders also have their own areas of expertise and have value to add to a project. An effective approach is to assume that everyone involved with your project has equal value and therefore should be treated with respect. Huet Landry suggests the concept of "Other Esteem", instead of "Self Esteem", where you treat the opinions of others as if they have more value than yours. With this approach your first reaction to another's idea will be most positive.

1.2: Agile Process Framework

What is NOT an Agile software development?

- Compressing the project schedule
- Eliminating all existing software development models
- Eliminating all documentation
- Writing code up to the last minute
- An excuse for doing nothing

What is NOT an Agile software development?

While there are many different definitions of what Agile Software Development is, there are reasons why each of the quoted sources view Agile in different ways. The term Agile is used to describe a software development approach that embodies the statements made in Agile Manifesto and follows the “12 Principles of Agile Methods”.
Many software development group says that they are using agile methods, but calling the development approach Agile doesn’t actually make it AGILE.
If you have implemented any of the above as part of your agile development approach, then you might need to rethink what agility actually is.

1.2: Agile Process Framework

Foundation of an Agile software development Method

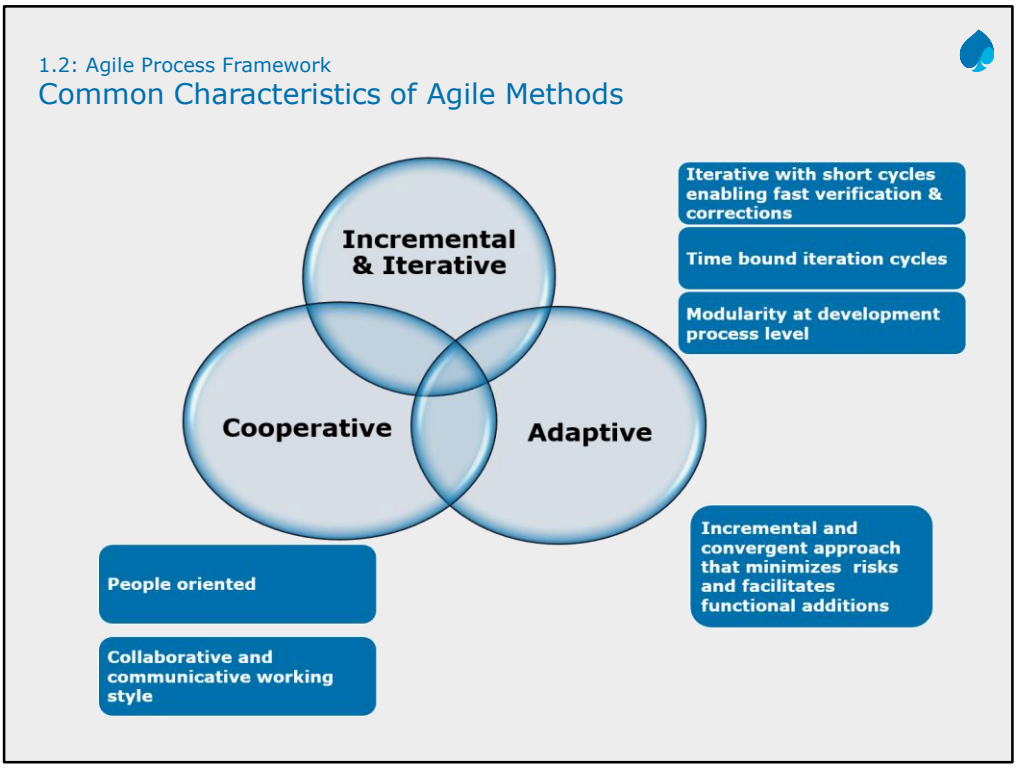
- Agility
- Change
- Planning
- Communication
- Learning
- Team

Foundation of an Agile software development Method

1. Agility – The ability to respond quickly.
2. Change – Is inevitable and important to be taken care of.
3. Planning – Creating small increments of the software system which is delivered as a series of time boxed iterations.
4. Communication – With the help of constant discussions & workshops aimed at information exchange between customer & team. This attribute is demonstrated via the daily stand-up meetings, collective iteration and release planning and the workshops held to flash out the user stories selected for each iteration.
5. Learning – Throughout the project cycle by analyzing and looking for the most appropriate ways to implement improvements based on the results of the previous iterations or release increments i.e. don't repeat the same behavior if it didn't result in the desired outcome.
6. Team – Authorized to take decisions and accountable for the release of the items committed to during the iteration planning.

1.2: Agile Process Framework

Common Characteristics of Agile Methods



1.2: Agile Process Framework

Agile Methods and Practices

- Scrum - Ken Schwaber, Jeff Sutherland, Mark Beedle
- Extreme Programming (XP) - Kent Beck, Eric Gamma, and others
- Dynamic System Development Method (DSDM) - Dane Faulkner And others
- Agile Unified Process (or Agile RUP) - Scott Ambler
- Feature Driven Development - Peter Coad and Jeff Deluca
- Lean Software Development - Mary and Tom Poppendieck
- Kanban - David Anderson

The term 'agile' is a philosophy and is a conceptual framework for undertaking software engineering projects. Most agile methods attempt to minimize risk by developing software in short time boxes, called iterations. Each iteration is like a miniature software project of its own, and includes all of the tasks necessary to release the mini increment of new functionality planning, requirements analysis, design, coding, testing and documentation. While an iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. Under this broad umbrella sits many more specific approaches.

Some of the well-known agile software development methods are given below.

1. Scrum - Ken Schwaber, Jeff Sutherland, Mark Beedle
2. Extreme Programming (XP) - Kent Beck, Eric Gamma, and others
3. Dynamic System Development Method (DSDM) - Dane Faulkner And others
4. Agile Unified Process (or Agile RUP) - Scott Ambler
5. Feature Driven Development - Peter Coad and Jeff Deluca
6. Lean Software Development - Mary and Tom Poppendieck
7. Kanban - David Anderson

1.2: Agile Process Framework

When to use Agile Model?

- This model can be followed when:
  - New changes must be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
  - To implement a new feature, the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
  - Unlike the Waterfall Model, in the agile model, limited planning is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This gives the customer the finished system they want or need.
  - Both system developers and stakeholders alike, find that they also get more freedom of time and options than if the software was developed in a more rigid, sequential manner. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

1.2: Agile Process Framework

Advantages of Agile Model

- Is a very realistic approach to software development
- Promotes teamwork and cross training
- Functionality can be developed rapidly and demonstrated
- Resource requirements are minimum
- Suitable for fixed or changing requirements
- Delivers early partial working solutions
- Good model for environments that change steadily
- Minimal rules, documentation easily employed
- Enables concurrent development and delivery within an overall planned context
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

1.2: Agile Process Framework

Disadvantages of Agile Model

- Not suitable for handling complex dependencies
- More risk of sustainability, maintainability and extensibility
- An overall plan, an agile leader and agile PM practice is a must without which it will not work
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction
- There is very high individual dependency, since there is minimum documentation generated
- Transfer of technology to new team members may be quite challenging due to lack of documentation

1.2: Agile Process Framework

Difference between Agile and Waterfall Model

Agile	Waterfall
• Software development lifecycle is carried out in the form of Sprints • Agile method proposes incremental and iterative approach to software design • It follows an incremental approach towards solution development • Agile methodology is known for its flexibility • Agile can be considered as a collection of many different projects	• Software development process is divided into distinct phases • Development of the software flows sequentially from start point to end point • It follows linear, sequential design approach towards software development • Being a traditional software development model, Waterfall exhibits characteristic of a structured model so most of the times it can be very rigid • Software development will be completed as one single project

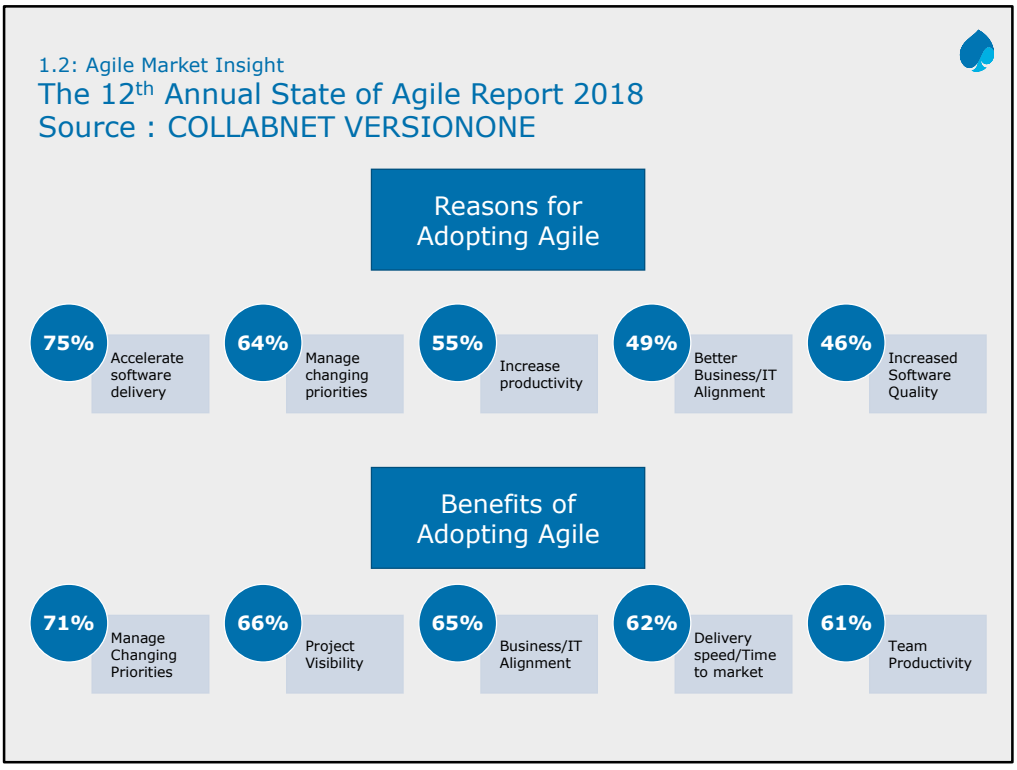
1.2: Agile Process Framework

Agile – Myths and Reality

Myth	Reality
• No Documentation	• Agile Documentation
• Undisciplined	• Requires great discipline
• No Planning	• Just-in-time (JIT) planning
• Not Predictable	• Far more predictable
• Is a Fad	• It's quickly becoming the norm
• Silver Bullet	• It requires skilled people
• RUP isn't agile	• RUP is as agile as you make it
• Not Fixed Price	• Agile provides stakeholders control over the budget, schedule, and scope

1.2: Agile Market Insight

The 12th Annual State of Agile Report 2018
Source : COLLABNET VERSIONONE



Summary

- In this lesson, you have learnt
  - Various conventional software development models like Waterfall Model, SDLC & V-Model
  - The traditional software development models like Waterfall Model, V-Model are classified into the heavyweight methodologies
  - These methodologies are based on progressive series of steps like requirements definition, design and architectural planning, development and testing
  - Every traditional software development model has its own advantages and disadvantages
  - We need to select the software development model which best suits our project requirement
  - Introduction to Agile
  - Agile Manifesto and Principles
  - Agile Values
  - Agile – Myths and Reality
  - Agile Market Insight

Add the notes here.