Agile Software Development (ASD) Lect-2

Prof. Daiwat Vyas

B.TECH SEMESTER 6TH CSE

Fundamentals Of Agile

- The Genesis of Agile
- Introduction and Background
- Agile Manifesto & Principles

The Genesis of Agile

- In the early 1990s, as PC computing began to proliferate in the enterprise, software development faced a crisis.

- At the time, it was widely referred to as "the application development crisis," or "application delivery lag."

- Industry experts estimated that the time between a validated business need and an actual application in production was about three years.

The Genesis of Agile

- The problem was, businesses moved faster than that, even 25 years ago.

- Within the space of three years, requirements, systems, and even entire businesses were likely to change.

- That meant that many projects ended up being cancelled partway through, and many of those that were completed didn't meet all the business's current needs, even if the project's original objectives were met.

The Genesis of Agile

- In certain industries, the lag was far greater than three years. In aerospace and defense, it could be 20 or more years before a complex system went into actual use.

- In an extreme but by no means unusual example, the Space Shuttle program, which operationally launched in 1982, used information and processing technologies from the 1960s.

- Highly complicated hardware and software systems were often designed, developed, and deployed in a time frame that spanned decades.

The Genesis of Agile

- Jon Kern, an aerospace engineer in the 1990s, became increasingly frustrated with these long lead times and with the decisions made early in a project that couldn't be changed later.

- "We were looking for something that was more timely and responsive," he notes, joining a growing number of those who felt that there had to be a better way to build software.
- He was one of 17 software thought leaders who started meeting informally and talking about ways to develop software more simply, without the process and documentation overhead of waterfall and other popular software engineering techniques of the time.

The Genesis of Agile

- Other industries were also undergoing transformation. It took the automotive industry six years or more to design a new car, and in the 1990s, that time was cut almost in half.

- For those products with a software development component, such as phone switches, autos, or aircraft, software was often an afterthought, mostly because software development didn't start until the hardware design was fixed in place.

- But building the software wasn't a priority for most product teams at the time.

The Genesis of Agile

- Birth of Agile:

- The frustrations around seemingly unproductive software development activities, which were shared by like-minded professionals, led to the now-famous Snowbird meeting in Utah in early 2001.

- But that wasn't the first time this particular group of software leaders had met. They had gathered the year before, at the Rogue River Lodge in Oregon in the spring of 2000.

The Genesis of Agile

- This group included Kern, Extreme Programming pioneers Kent Beck and Ward Cunningham, Arie van Bennekum, Alistair Cockburn, and twelve others, all well known today in the agile community.

- In particular, these thought leaders sought ways to quickly build working software and get it into the hands of end users.

- This fast delivery approach provided a couple of important benefits.

- First, it enabled users to get some of the business benefits of the new software faster.
- Second, it enabled the software team to get rapid feedback on the software's scope and direction.

The Genesis of Agile

- Rapid feedback and willingness to change turned out to be the key features of the agile movement.

- If the software team isn't confident in understanding what the user needs, it delivers a first approximation and then listens to feedback.

- But little is set in stone at the beginning of the project.

Introduction to Agile

- Why Rapid Software Development needed?
- Need to react to changes more quickly

- How?

  - Goal - Deliver working software quickly
    - Compromise - less functionality in a delivery, not lower quality
    - Less documentation
  - Interleave
    - Specification, design and implementation are inter-leaved
  - Deliver small versions and get user (stakeholder) input

Introduction to Agile from 12 Jan 2024

- What is Agile?
- Agile --readiness for motion, nimbleness, activity, dexterity in motion

- Agility

  The ability to both create and respond to change in order to profit in a turbulent business environment

  - Companies need to determine the amount of agility they need to be competitive

- Chaordic
  - Exhibiting properties of both ***cha***os and ***ord***er
    - The blend of chaos and order inherent in the external environment and in people themselves, argues against the prevailing wisdom about predictability and planning
    - Things get done because people adapt, not because they slavishly follow processes

Introduction to Agile

- **What is Agile?**
- **Dictionary meaning:** Agile is the ability to create and respond to changes to succeed in an uncertain environment.
  - i.e respond to changes in an apt manner

- **What is Agile Software Development (ASD)?**
- ASD is a collection of Agile methodologies that promote adaptive planning, development, improvements, and delivery.
- It also means a time-boxed period of time to complete work.

Introduction to Agile

- **Popular ASD Methodologies**

- Scrum
- XP – Extreme programming
- DSDM – Dynamic Systems Development Method
- FDD – Feature-Driven Development
- Crystal
- LSD – Lean Software Development
- KANBAN

Introduction to Agile

- **Agile software development** is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.

- Software developed during one unit of time is referred to as an iteration, which may last from one to four weeks.

- Agile methods also emphasize working software as the primary measure of progress

Introduction to Agile

- **Agile** development is a different way of managing IT development teams and projects.
- Agile Software Development is a **set of software development methods** in which **requirements and solutions develop through collaboration between self-organizing, adaptive planning, early delivery, continuous improvement, and encourages rapid and flexible response to change.**
- Agile software development is a conceptual framework for software engineering that promotes development iterations throughout the life-cycle of the project.

Introduction to Agile

- Characteristics of Agile Software Development
  - -- Light Weighted methodology
  - -- Small to medium sized teams
  - -- vague and/or changing requirements
  - -- vague and/or changing techniques
  - -- Simple design
  - -- Minimal system into production

Introduction to Agile

- **Advantages of ASD**

- The Agile approach encourages rapid and flexible response to changes.

- Teams that use Agile are well-equipped to respond to changes throughout the development cycle as adaptability is central to the conceptual framework.

- Agile software development promotes a different way of thinking about how to create things and evolve processes to deliver continuous improvement. Here all the decisions are made together as a team.

Agile Manifesto & Principles

- The Agile Manifesto is a set of guiding values and principles for software development that prioritize individuals and interactions, working solutions, and customer collaboration over processes and tools.

- It was created by a group of software developers who met in 2001 and aimed to address the shortcomings of traditional software development methodologies.

- The **Agile Manifesto** consists of **four key values** and **twelve principles.**

Agile Manifesto & Principles

- The Agile Manifesto is a set of guiding values and principles for software development that prioritize individuals and interactions, working solutions, and customer collaboration over processes and tools.

- It was created by a group of software developers who met in 2001 and aimed to address the shortcomings of traditional software development methodologies.

- The **Agile Manifesto** consists of **four key values** and **twelve principles.**

Agile Manifesto & Principles

- 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.

Agile Manifesto & Principles

- **Agile Manifesto Values:**
- Individuals and Interactions over Processes and Tools:
  - Emphasizes the importance of people and effective communication in the development process.
- Working Solutions over Comprehensive Documentation:
  - Prioritizes functional software over extensive documentation, recognizing that the focus should be on delivering a working product.
- Customer Collaboration over Contract Negotiation:
  - Encourages active collaboration with customers throughout the development process, adapting to changing requirements and needs.
- Responding to Change over Following a Plan:
  - Values the ability to adapt and respond to changing requirements over strictly adhering to a predetermined plan.

Principles behind the Agile Manifesto

The following **principles** are followed in ASD:

- Satisfy the Customer Through Early and Continuous Delivery of Valuable Software
  - Deliver working software frequently, with a preference for shorter timescales.
- Welcome Changing Requirements, Even Late in Development
  - Embrace changes in requirements, even if they occur late in the development process, to provide the customer with a competitive advantage.
- Deliver Working Software Frequently, with a Preference for the Shortest Timescale
  - Strive to deliver a functional product as early as possible and continue to deliver incremental updates frequently.

Principles behind the Agile Manifesto

The following **principles** are followed in ASD:

- Collaborate with Customers Throughout the Development Process
  - Actively involve customers and stakeholders throughout the development process to ensure the product meets their needs.

- Build Projects around Motivated Individuals, Give Them the Environment and Support They Need, and Trust Them to Get the Job Done
  - Foster a supportive environment that empowers and trusts motivated individuals, recognizing them as the primary drivers of successful projects.

Principles behind the Agile Manifesto

The following **principles** are followed in ASD:

- Use Face-to-Face Communication Wherever Possible
  - Promote direct, face-to-face communication to enhance understanding and collaboration among team members.

- Working Software Is the Primary Measure of Progress
  - Assess progress primarily through the delivery of functional software, rather than relying solely on documentation or other metrics.

- Maintain a Sustainable Pace of Work for the Development Team
  - Promote a work environment that allows the development team to maintain a sustainable pace, avoiding burnout and promoting long-term productivity.

Principles behind the Agile Manifesto

The following **principles** are followed in ASD:

- Strive for Technical Excellence and Good Design
  - Emphasize the importance of technical excellence and good design to enhance the agility and maintainability of the software.
- Simplicity—the Art of Maximizing the Amount of Work Not Done—is Essential
  - Prioritize simplicity and efficiency, focusing on delivering the necessary features and functionality without unnecessary complexity.
- Self-Organizing Teams Produce the Best Architectures, Designs, and Requirements
  - Encourage self-organizing teams, as they are best suited to determine the most effective architecture, design, and requirements for a project.

Principles behind the Agile Manifesto

The following **principles** are followed in ASD:

- At Regular Intervals, Reflect on How to Become More Effective, Then Tune and Adjust the Behavior Accordingly
  - Regularly review and reflect on the development process to identify opportunities for improvement, and adjust practices accordingly.

Summary of 12 Principles of ASD

1. Customer Satisfaction
2. Welcome Change
3. Deliver a Working Software
4. Collaboration
5. Motivation
6. Face-to-face Conversation
7. Measure the Progress as per the Working Software
8. Maintain Constant Pace
9. Monitoring
10. Simplicity
11. Self-organized Teams
12. Review the Work Regularly

When to use ASD approach? Factors??

- When best to use Agile methodology – this question depends on many factors:
  - including the number of available resources (Agile projects are quite demanding)
  - type of development project (some of them work best with traditional development methods)
  - trained professionals (teams members are crucial for a successful Agile project), etc

When to use ASD approach? Factors??

- Uncertain or rapidly changing requirements:
  - Agile is well-suited for projects where the requirements are expected to evolve or change frequently.
  - It allows teams to adapt quickly to changing needs and priorities.

- Complex projects:
  - Projects with a high level of complexity or those with a lot of unknowns can benefit from Agile.
  - The iterative nature of Agile allows teams to break down complex tasks into smaller, manageable pieces and address challenges as they arise.

When to use ASD approach? Factors??

- Customer involvement and feedback are critical:
  - If customer involvement and feedback are essential throughout the development process, Agile is a good fit.
  - Regular iterations and demos allow customers to provide input, ensuring that the final product aligns with their expectations.

- Short time-to-market requirements:
  - Agile promotes incremental development, allowing for the delivery of functional increments in short cycles.
  - This can be advantageous when there is a need for a quick time-to-market or when releasing a minimum viable product (MVP) for initial user feedback.

When to use ASD approach? Factors??

- Highly collaborative team environment:
  - Agile relies on collaboration and communication among team members.
  - If the team is cross-functional, self-organizing, and can work closely together, Agile methodologies can be effective in fostering collaboration and quick decision-making.

- Frequent reassessment of priorities:
  - Agile methods, such as Scrum, involve regular sprint planning meetings where team priorities are reassessed.
  - If project priorities are likely to change frequently, Agile provides a framework for adapting to these changes efficiently.

When to use ASD approach? Factors??

- Continuous improvement mindset:
  - Agile encourages a culture of continuous improvement.
  - If the team values reflection on processes and outcomes and is committed to making adjustments based on lessons learned, Agile practices can help refine and enhance the development process over time.

- Small to medium-sized teams:
  - Agile methodologies are often more effective with smaller, cross-functional teams.
  - Large projects may benefit from scaling frameworks, but small to medium-sized teams typically find Agile principles more manageable and applicable.

When to use ASD approach? Factors??

- Flexible project scope:
  - If the project has a flexible scope and can be delivered incrementally, Agile is a good fit.
  - It allows for changes in scope and priorities without disrupting the entire development process.

When to use ASD approach? Factors??

- It's important to note that while Agile is suitable for many scenarios, it may not be the best fit for every project.
- Factors such as organizational culture, project size, and the nature of the product should also be considered when deciding on the development approach.
- Additionally, hybrid approaches that combine Agile with other methodologies may be appropriate in some situations.

When not to use ASD approach?

- The project is not very urgent, too complex or novel.
- The team is not self-organizing and lacks professional developers/experience.
- The customer requires neat documentation of each development cycle.
- The customer requires approvals at each stage of development.
- The Software Development organization does not invest in spreading Agile practices among the team.

When not to use ASD approach?

- The project is not very urgent, too complex or novel.
  - Agile methodology is quite demanding, so there is no need to use it for simple or typical projects.
  - Simple/Typical projects can be easily accomplished with traditional Waterfall methodology. Long cycles, clear development goals, and typical cycles – all of these aspects will make your life easier with traditional methods.

When not to use ASD approach?

- The project is not very urgent, too complex or novel. (contd...)
  - Agile methods allow you to create a lasting, well-organized software development process, highly adaptable to the changing requirements and environment.
  - If you have a clear goal and enough time to accomplish your project, this is when you should consider not using Agile development methods

When not to use ASD approach?

- The team is not self-organizing and lacks professional developers/experience.

  - A well-trained team, aware of Agile principles and practices, is one of the most important aspects of any Agile project.

  - The agile process requires a lot of crucial decisions to be taken by the team members during the project.

  - And if your developers are not experienced and responsible enough, this may ruin the whole project.

When not to use ASD approach?

- The team is not self-organizing and lacks professional developers/experience. (contd...)

  - Do not forget that Agile methods require constant interaction among all stakeholders – development team, testing unit, management, and the customer.

  - If one of the teams shows poor performance, this will definitely affect the overall result.

  - If you have doubts as to your development team – this is when you should not use Agile software development, since it can backfire at you at any stage of the process.

When not to use ASD approach?

- The customer requires neat documentation of each development cycle.
  - In many cases, your customer will prefer your company to submit detailed documentation of each phase of the project.
  - These requirements mostly hold for infrastructure applications, which are designed to support a process for a long period of time.
  - Unlike business process applications, which are continuously adapted to the changeable business environment, infrastructure applications will remain unchanged for quite some time.

NOTE

- Infrastructure application is a type of enterprise software or program specifically designed to help business organizations perform basic tasks such as workforce support, business transactions and internal services and processes.

- The most common examples of infrastructure software are database programs, email and other communication software and security applications.

When not to use ASD approach?

- The customer requires approvals at each stage of development.
  - Maybe your customer is a vivid adherent of Waterfall methods and demands you to deliver software pieces for approval at each development cycle.
  - Or maybe you develop software for a bureaucratic organization with strict rules or procedures.
  - Anyway, if your deliverables have to go through a chain of tollgates at every stage, you need to forget about Agile methods.
  - These bottlenecks will kill all the momentum, so crucial for a successful Agile project.

When not to use ASD approach?

- The Software Development organization does not invest in spreading Agile practices among the team.

  - Some organizations prefer to work with customers who stick to the traditional development practices.

  - And when the time for Agile arrives, it turns out that the communication links and resource base for a successful Agile project are lacking.

  - Try to avoid resorting to Agile methods with a poor foundation, as it will negatively reflect upon your work.

When not to use ASD approach?

- Remember that-

- There is always a trade-off while you take a decision to use ASD and there are always ways to cater the negatives/shortfalls.

How Agile differs from waterfall approach?

- ??????

How Agile differs from waterfall approach?

- **Project Lifecycle:**
- **Waterfall:** Follows a linear and sequential approach.
- The project is divided into distinct phases (requirements, design, implementation, testing, deployment, and maintenance), and each phase must be completed before moving on to the next.

- **Agile:** Embraces an iterative and incremental approach.
- The project is divided into small, manageable increments, with each iteration delivering a potentially shippable product.

How Agile differs from waterfall approach?

- **Flexibility and Change:**
- **Waterfall:** Changes to requirements are often difficult to accommodate once the project has started.
- The entire project must be revisited if there are significant changes.

- **Agile:** Welcomes changes even late in the development process.
- It allows for frequent reassessment and adaptation, allowing teams to respond to changing requirements and priorities.

How Agile differs from waterfall approach?

- **Customer Involvement:**
- **Waterfall:** Customer involvement is typically at the beginning and the end of the project.
- The customer provides requirements at the start and accepts the final product at the end.

- **Agile:** Encourages continuous customer involvement throughout the development process.
- Regular feedback from customers is sought to ensure that the product meets their evolving needs.

How Agile differs from waterfall approach?

- **Customer Involvement:**
- **Waterfall:** Customer involvement is typically at the beginning and the end of the project.
- The customer provides requirements at the start and accepts the final product at the end.

- **Agile:** Encourages continuous customer involvement throughout the development process.
- Regular feedback from customers is sought to ensure that the product meets their evolving needs.

How Agile differs from waterfall approach?

- **Delivery Time:**
- **Waterfall:** Longer delivery times, as the entire project must be completed before delivering the final product.
- **Agile:** Shorter delivery times, with the ability to release incremental versions of the product at the end of each iteration.

How Agile differs from waterfall approach?

- **Documentation:**
- **Waterfall:** Requires comprehensive documentation at each phase of the project.
- **Agile:** Values working software over extensive documentation, although sufficient documentation is still important.

How Agile differs from waterfall approach?

- **Team Collaboration:**

- **Waterfall:** Team collaboration is limited, with each phase handed off to the next team once completed.

- **Agile:** Encourages collaboration among team members throughout the entire development process.

How Agile differs from other modern day project development approaches like spiral, iterative etc.?

- **Agile:**
- **Principles:** Agile is based on the Agile Manifesto, which values individuals and interactions, working solutions, customer collaboration, and responding to change.
- **Process:** Agile promotes iterative and incremental development, with a focus on delivering a minimum viable product (MVP) quickly and then continuously iterating based on feedback.
- **Flexibility:** Agile is highly adaptive to changing requirements and customer feedback throughout the development process.
- **Roles:** Agile encourages collaboration among cross-functional teams and embraces customer involvement throughout the project.

How Agile differs from other modern day project development approaches like spiral, iterative etc.?

- **Spiral:**

- **Risk Management:** The Spiral model emphasizes risk management and addresses potential risks early in the development process.
- **Iterations:** The model involves iterative cycles, with each iteration going through planning, risk analysis, engineering, and evaluation.
- **Phases:** Spiral typically has multiple phases, such as determining objectives, risk analysis, development and testing, planning the next iteration, and repeating the process.
- **Documentation:** Documentation is extensive and is produced at every phase of the spiral.

How Agile differs from other modern day project development approaches like spiral, iterative etc.?

- **Iterative:**
- **Repetitive Cycles:** Iterative development involves repeating cycles of planning, designing, implementing, and testing.
- **Incremental Development:** Similar to Agile, iterative development focuses on delivering a portion of the complete system in each iteration.
- **Feedback:** Iterative development encourages obtaining feedback from stakeholders at the end of each iteration and incorporating it into subsequent iterations.
- **Phases:** It may include phases like requirements, design, implementation, and testing, with each phase being revisited in each iteration.

How Agile differs from other modern day project development approaches like spiral, iterative etc.?

- **Key Differences:**
- **Flexibility:** Agile is known for its flexibility and adaptability to changing requirements, whereas Spiral focuses on managing risks and Iterative emphasizes repetitive cycles.
- **Customer Involvement:** Agile places a strong emphasis on continuous customer involvement and collaboration, whereas Spiral and Iterative may have less frequent customer interactions.
- **Risk Management:** Spiral explicitly integrates risk management into the development process, while Agile and Iterative also address risks but may not follow a structured risk management process like Spiral.
- **Documentation:** Spiral tends to produce extensive documentation at each phase, while Agile values working solutions over comprehensive documentation, and Iterative falls somewhere in between.

How Agile differs from other modern day project development approaches like spiral, iterative etc.?

- **Key Differences:**
- **Flexibility:** Agile is known for its flexibility and adaptability to changing requirements, whereas Spiral focuses on managing risks and Iterative emphasizes repetitive cycles.
- **Customer Involvement:** Agile places a strong emphasis on continuous customer involvement and collaboration, whereas Spiral and Iterative may have less frequent customer interactions.
- **Risk Management:** Spiral explicitly integrates risk management into the development process, while Agile and Iterative also address risks but may not follow a structured risk management process like Spiral.
- **Documentation:** Spiral tends to produce extensive documentation at each phase, while Agile values working solutions over comprehensive documentation, and Iterative falls somewhere in between.

Thank you