UNIT 5

Agile Software Development: Introduction to agile methods, Agile development techniques, Agile project management and scaling agile methods.

Kanban, Flow, and Constantly Improving: The Principles of Kanban, Improving Your Process with Kanban, Measure and Manage Flow, Emergent Behavior with Kanban.

The Agile Coach: Coaches Understand Why People Don't Always Want to Change, Coaches Understand How People Learn, Coaches Understand What Makes a Methodology Work, The Principles of Coaching.

SECTION:

Agile Software Development: Introduction to agile methods, Agile development techniques, Agile project management and scaling agile methods.

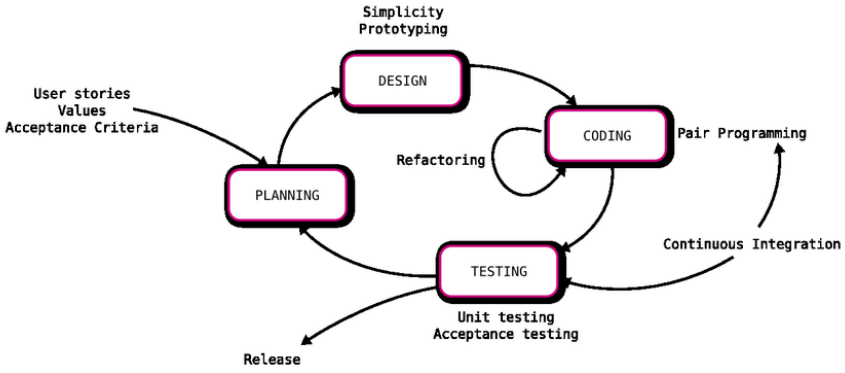
1	**12 Principles of Agile Software:** 1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software. 2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. 3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. 4. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. 5. Businesspeople and developers must work together daily throughout the project. 6. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. 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.
2	**Identify the common characteristics between the agile teams and other successful teams that build software using a waterfall process.** 1. Good communication, because the teams that were successful in a company that mandated waterfall were the ones that consistently talked to their users, managers, and executives throughout the project. 2. Good practices, especially ones like code reviews and automated testing, which are aimed at finding bugs as early as possible in the process. 3. Defect prevention requires teams to actively think about how those bugs got into the code in the first place.

	4. Drawers full of documentation that have rarely been opened, because the people on the team understand that the act of writing down the plan—and the questions that get asked during that planning—is more important than mindlessly sticking to it afterward.
3	**Appraise the role and significance of Product Owner in Scrum teams.** 1. Understand what the company needs most, and bring that knowledge back to the team. 2. Understand what software features the team can potentially deliver. 3. Figure out which features are more valuable to the company, and which are less valuable. 4. Work with the team to figure out which features are easier to build, and which are harder. 5. Use that knowledge of value, difficulty, uncertainty, complexity, etc. to help the team choose the right features to build in each sprint. 6. Bring that knowledge back to the rest of the company, so they can do what they need to do to prepare for the next release of the software.

4	# Agile Software Development: Introduction to Agile Methods
	
	## What is Agile?
	
	• Agile is a mindset and set of principles for software development focused on **iterative development, collaboration, flexibility, and delivering working software frequently**.
	• It emerged as a response to the shortcomings of traditional "waterfall" methods that were often rigid, slow, and unable to cope with changing requirements.
	
	## The Agile Manifesto
	
	• Created in 2001 by a group of software practitioners.
	• Values:
	  ○ **Individuals and interactions** over processes and tools
	  ○ **Working software** over comprehensive documentation
	  ○ **Customer collaboration** over contract negotiation
	  ○ **Responding to change** over following a plan
	• These values emphasize flexibility, communication, and delivering real value.
	
	## Agile Principles (12 key principles)
	
	• Deliver working software frequently (weeks rather than months)
	• Welcome changing requirements, even late in development
	• Business people and developers work together daily
	• Build projects around motivated individuals
	• Face-to-face conversation is the best form of communication
	• Working software is the primary measure of progress
	• Sustainable development pace
	• Continuous attention to technical excellence and good design
	• Simplicity—the art of maximizing work not done—is essential
	• Self-organizing teams produce the best architectures and designs

	• Regular reflection on how to become more effective, adjusting behavior accordingly **Overview of Agile Methods** • Scrum, Extreme Programming (XP), Lean Software Development, Kanban • Each method implements Agile principles differently with unique practices and focuses.
5	# Agile Development Techniques **Iterative and Incremental Development** • Work is broken into small, manageable chunks called iterations or sprints (typically 1-4 weeks). • Each iteration results in a potentially shippable product increment. • Feedback is gathered early and often, allowing for course correction. **User Stories and Backlogs** • Requirements are expressed as **user stories**: short descriptions of functionality from the user's perspective. • Stories are collected in a **product backlog**, prioritized by business value. **Planning and Estimation** • Agile uses relative estimation techniques like story points or ideal days. • Planning is done at multiple levels: release planning and iteration/sprint planning. • Planning is adaptive and revisited regularly. **Continuous Integration and Test-Driven Development (TDD)** • Continuous integration ensures that code is frequently merged and tested. • TDD involves writing automated tests before the code, improving quality and design. **Pair Programming and Collective Code Ownership (XP practices)** • Two developers work together on the same code, improving knowledge sharing and code quality. • Everyone can modify any part of the codebase, encouraging shared responsibility. **Refactoring** • Continuously improving the internal structure of the code without changing its external behavior to keep the codebase healthy.

	Daily Standups (Scrum) • Short daily meetings to coordinate, discuss progress, and surface impediments.
5	# Agile Project Management **Role of the Product Owner** • Represents stakeholders and customers. • Maintains and prioritizes the product backlog. • Makes decisions about features and release scope. **Role of the Scrum Master (or Agile Coach)** • Facilitates the team's Agile process. • Removes impediments and shields the team from distractions. • Coaches the team and organization in Agile practices. **The Agile Team** • Cross-functional and self-organizing. • Responsible for delivering the product increments. • Encouraged to collaborate and make collective decisions. **Planning and Tracking Progress** • Use of **burndown charts** to visualize remaining work. • Regular iteration reviews and retrospectives. • Emphasis on **empirical process control**: inspect and adapt based on real data. **Agile Contracts and Stakeholder Management** • Contracts may be flexible, focusing on collaboration and value rather than fixed scope. • Frequent demos and feedback loops to ensure alignment.
6	# Scaling Agile Methods **Challenges of Scaling** • Agile methods work well with small teams, but larger organizations need to coordinate multiple teams. • Challenges include coordination, dependencies, maintaining communication, and preserving Agile values. **Popular Scaling Frameworks** • **SAFe (Scaled Agile Framework)**: Provides roles, artifacts, and ceremonies for scaling Agile across large enterprises.

	• **LeSS (Large-Scale Scrum)**: Extends Scrum principles for multiple teams working on the same product. • **Scrum of Scrums**: A technique for coordinating multiple Scrum teams through representatives. **Coordination Techniques** • Integration teams or roles. • Shared product backlogs with dependencies clearly mapped. • Synchronization meetings across teams. • Use of Agile project management tools for visibility. **Lean and Kanban for Scaling** • Kanban boards visualize workflow across teams. • Focus on managing flow, limiting work in progress (WIP), and improving cycle time. • Lean principles applied to reduce waste and increase value delivery.
7	XP: the XP life cycle and its iterative, feedback-driven process • XP is one of the core Agile methodologies alongside Scrum, Lean, and Kanban. • XP-specific practices - pair programming, test-driven development (TDD), continuous integration, simple design, and frequent releases—all key parts of the XP life cycle. • XP is more team-focused, elements of its planning, iteration cycles, **Extreme Programming (XP) Life Cycle Steps and Practices:** **XP Life Cycle Overview**  XP emphasizes delivering high-quality software through **short, iterative development cycles** with frequent feedback. The XP life cycle is designed to maximize communication, simplicity, feedback, and courage. **1. Exploration Phase** • The team collaborates with customers to gather initial requirements in the form of **user stories**.

- The goal is to understand what needs to be built and prioritize features.
- Team members experiment with new technologies or architectural ideas if needed.
- Early tests and prototypes may be created.

2. Planning Phase

- The team estimates user stories using relative sizing (often story points or ideal days).
- Stories are prioritized based on business value and risk.
- A release plan is created, mapping stories to iterations.
- Iterations typically last 1-2 weeks.

3. Iteration to Release Phase

- Development proceeds in short iterations (1-2 weeks).
- Each iteration delivers a potentially shippable increment.
- Practices emphasized during iterations include:
    - **Pair Programming:** Two developers work together at one workstation to improve code quality.
    - **Test-Driven Development (TDD):** Writing automated tests before code to ensure correctness.
    - **Continuous Integration:** Frequently integrating code into the mainline to detect issues early.
    - **Refactoring:** Continuously improving the design and structure of existing code without changing behavior.
    - **Simple Design:** Keeping the system design as simple as possible to meet current requirements.
    - **Collective Code Ownership:** Anyone on the team can modify any part of the codebase.
- Frequent communication with the customer to refine stories and accept completed features.

4. Productionizing Phase

- Focus on stabilizing the system for production release.
- Additional testing, bug fixing, and performance tuning.
- Final documentation and deployment preparations.
- Addressing any last-minute integration issues.

5. Maintenance Phase

- Ongoing support and incremental improvements.
- New user stories added and prioritized for future iterations.
- Continuous refactoring and adaptation to changing requirements.

7

Topic	Key Points to Remember
Agile Introduction	- Agile values individuals & interactions over processes.- Emphasizes working software and customer collaboration.- Welcomes changing requirements.- Iterative, incremental delivery.

Agile Development Techniques	- Use iterations/sprints (1-4 weeks) to deliver small increments.- Capture requirements as user stories.- Estimate work with story points.- Practice continuous integration and TDD.- Pair programming and collective ownership improve quality.- Daily standups for team communication.- Refactor regularly to improve code.
Agile Project Management	- Product Owner prioritizes backlog and represents stakeholders.- Scrum Master facilitates process and removes blockers.- Teams are cross-functional and self-organizing.- Use burndown charts to track progress.- Regular reviews and retrospectives improve process.- Agile contracts favor collaboration over fixed scope.
Scaling Agile Methods	- Scaling requires coordination across multiple teams.- Frameworks: SAFe, LeSS, Scrum of Scrums.- Synchronize teams with integration roles and shared backlogs.- Lean/Kanban help visualize workflow and limit WIP.- Focus on reducing waste and improving cycle time at scale.

SECTION:

Kanban, Flow, and Constantly Improving: The Principles of Kanban, Improving Your Process with Kanban, Measure and Manage Flow, Emergent Behavior with Kanban.

1

1. Kanban: Principles and Overview

What is Kanban?

- Kanban is an Agile method focused on **visualizing work**, **limiting work in progress (WIP)**, and **managing flow**.
- Originated in manufacturing (Toyota Production System) and adapted for software development and knowledge work.
- Emphasizes **continuous delivery** and **evolutionary change** rather than radical transformation.

Core Principles of Kanban

- **Visualize the workflow:** Use a Kanban board to display all work items and stages clearly.
- **Limit Work In Progress (WIP):** Set explicit limits on how many items can be in any one stage to avoid bottlenecks.
- **Manage flow:** Monitor and improve how work moves through the process.
- **Make process policies explicit:** Define clear rules for how work moves through the stages.
- **Implement feedback loops:** Regular reviews and retrospectives to refine process.
- **Improve collaboratively, evolve experimentally:** Encourage continuous, incremental improvement based on real data.

Kanban Board Components

- **Columns:** Represent stages of the workflow (e.g., To Do, In Progress, Testing, Done).

- **Cards:** Represent work items (features, bugs, tasks).
- **Swimlanes:** Optional horizontal lanes to organize cards by type, priority, or team.

Kanban is less about fixed phases and more about continuous flow, but you can think of the Kanban life cycle as a flow of work items moving through defined stages on a Kanban board. The life cycle emphasizes **visualization, flow management, and continuous improvement**.

Kanban Life Cycle

1. Backlog or Request Stage

- Work items (features, bugs, tasks) are collected here.
- Items are prioritized but not yet started.
- Items wait until pulled into the workflow.

2. Ready / To Do

- Work items ready to be picked up.
- Explicit policies may define what "ready" means.

3. In Progress

- Work actively being done.
- WIP (Work-In-Progress) limits control how many items can be here simultaneously.
- Team members pull work based on capacity.

4. Review / Testing

- Work moves to review or testing stages.
- May include code review, QA testing, or stakeholder feedback.
- Bottlenecks often appear here if not managed well.

5. Done / Completed

- Work items that have met the definition of done.
- Ready for release or deployment.

6. Continuous Improvement

- Regular retrospectives and metrics (like cycle time, throughput).
- Identify bottlenecks and experiment with process improvements.
- Adjust WIP limits and policies as needed.

Key Principles Reflected in the Life Cycle

- **Visualize the workflow** using a Kanban board.
- **Limit Work-In-Progress** to avoid overloading the team.
- **Manage and improve flow** through measurement and adjustment.
- **Make process policies explicit** to improve clarity.
- **Implement feedback loops** (daily standups, retrospectives).
- **Continuously evolve** based on data and team learning.

2

Improving Your Process with Kanban

Evolutionary Change

- Kanban encourages **incremental improvements** without disrupting existing processes.
- Start with current workflow, visualize it, and gradually introduce improvements.
- Avoid forcing big changes; instead, evolve processes based on observed bottlenecks and pain points.

Identifying Bottlenecks

- By visualizing workflow and limiting WIP, teams can quickly spot stages where work piles up.
- Bottlenecks slow down flow and increase cycle time.
- Once identified, teams experiment with solutions (e.g., adjusting WIP limits, adding resources).

Policies and Agreements

- Make explicit agreements on how and when work is pulled or pushed.
- Define entry and exit criteria for each stage.
- Helps reduce ambiguity and improve process consistency.

Classes of Service

- Kanban can include different priority classes for work (e.g., expedited, standard, low priority).
- Helps manage flow and resource allocation based on urgency.

3

Measure and Manage Flow

Key Flow Metrics

- **Cycle Time:** Time taken for a work item to move from start to finish.
- **Lead Time:** Time from when a request is made until it is completed (includes waiting).
- **Throughput:** Number of items completed in a given time frame.

- **Work in Progress (WIP):** Number of work items currently active.

Using Metrics to Improve

- Monitor cycle and lead times to identify delays.
- Aim to reduce WIP limits to improve focus and decrease multitasking.
- Track throughput to measure team productivity.
- Use **cumulative flow diagrams** to visualize the number of items in each stage over time.

Managing Flow

- Smooth flow leads to predictable delivery and faster turnaround.
- Use WIP limits and prioritization to reduce task switching and overload.
- Flow-based management focuses on **pulling work** based on team capacity rather than pushing work based on fixed plans.

4

Emergent Behavior with Kanban

Self-Organization and Team Dynamics

- Kanban fosters a culture of **collaborative problem solving**.
- Teams become empowered to adjust policies and limits as they learn what works best.
- Feedback loops and transparency promote trust and ownership.

Continuous Improvement (Kaizen)

- Regular retrospectives and data-driven experiments lead to small, ongoing process enhancements.
- Encourages a mindset of **learning through doing** and adapting to changing needs.

Avoiding Common Pitfalls

- Don’t ignore team input or ignore data from metrics.
- Avoid setting WIP limits too low or too high; tune them based on team capacity.
- Resist temptation to micromanage; instead, coach and support the team’s self-organization.

Kanban as a Tool, Not a Silver Bullet

- Kanban supports evolutionary change but doesn’t prescribe all solutions.
- Success depends on the team’s commitment to transparency, communication, and continuous learning.

5

Topic	Key Points to Remember
Kanban Principles	Visualize work, limit WIP, manage flow, explicit policies, feedback loops, continuous improvement

Improving with Kanban	Incremental changes, identify bottlenecks, explicit policies, classes of service, evolve process based on data
Measure and Manage Flow	Key metrics: cycle time, lead time, throughput, WIP; use cumulative flow diagrams; smooth flow improves predictability
Emergent Behavior	Team self-organization, continuous improvement (Kaizen), data-driven experiments, avoid micromanagement

SECTION: The Agile Coach: Coaches Understand Why People Don't Always Want to Change, Coaches Understand How People Learn, Coaches Understand What Makes a Methodology Work, The Principles of Coaching.

1

Coaches Understand Why People Don't Always Want to Change

Resistance to Change

- Change triggers **uncertainty, fear, and discomfort** because it disrupts established routines and habits.
- People often prefer the known—even if imperfect—over unfamiliar territory.
- Resistance may come from:
  - Fear of failure or incompetence in the new process.
  - Loss of control or status.
  - Lack of trust in leadership or the change itself.
  - Unclear benefits or poor communication of why change is needed.

Psychological and Social Factors

- Change affects personal identity and group dynamics.
- People may fear social rejection or conflict if they adopt new ways.
- Change requires people to **unlearn** old habits and develop new skills, which takes time and effort.

Role of the Coach

- Recognize and empathize with resistance instead of forcing change.
- Build trust by listening and understanding individual concerns.
- Provide support and create safe environments for experimentation.
- Use small, incremental changes to reduce anxiety.
- Help people see the personal and organizational value in the change.

Benefits of having an agile coach in an organization

1. Guiding Teams Through Agile Adoption

An Agile Coach plays a pivotal role in helping teams transition from traditional methodologies to Agile practices. They assist in understanding and implementing Agile frameworks like Scrum, XP, Lean, and Kanban, ensuring that teams grasp not just the "how" but also the "why"

behind Agile principles. This guidance fosters a deeper commitment to Agile values and practices.

2. Facilitating Continuous Improvement

Coaches encourage a culture of continuous improvement by guiding teams through regular retrospectives and feedback loops. They help identify areas for enhancement, promote experimentation, and support teams in refining their processes to achieve higher efficiency and effectiveness.

3. Enhancing Team Collaboration and Communication

Effective communication and collaboration are at the heart of Agile methodologies. Agile Coaches facilitate open dialogues, encourage transparency, and help resolve conflicts, thereby strengthening team dynamics and fostering a collaborative environment that drives success.

4. Aligning Practices with Organizational Goals

An Agile Coach ensures that the team's Agile practices align with the organization's strategic objectives. By understanding both the team's capabilities and the organization's goals, they help bridge gaps, ensuring that Agile practices contribute to achieving broader business outcomes.

5. Building a Sustainable Agile Culture

Beyond implementing Agile practices, an Agile Coach works to instill an Agile mindset throughout the organization. They promote values such as flexibility, responsiveness, and customer-centricity, helping to cultivate a culture that supports long-term agility and adaptability.

Purpose of an Agile Coach

- **Facilitate Agile Adoption:** The Agile Coach's primary purpose is to help teams and organizations adopt Agile methodologies effectively. They guide people through the transition from traditional development models to Agile frameworks, ensuring a clear understanding of Agile principles, values, and practices.
- **Enable Behavioral and Cultural Change:** Agile transformation is not just about processes but also about people and culture. The coach supports individuals and teams in embracing new mindsets, overcoming resistance to change, and fostering a collaborative and transparent culture.
- **Support Learning and Growth:** Agile Coaches foster an environment where continuous learning and improvement are encouraged. They help teams reflect on their work, experiment with new approaches, and develop better ways of collaborating and delivering value.
- **Remove Impediments and Facilitate Collaboration:** Coaches help identify and remove obstacles that block team progress. They facilitate communication and collaboration within and across teams and with stakeholders to enhance productivity.
- **Empower Teams to Self-Organize:** A key goal of the Agile Coach is to nurture teams' autonomy and ability to self-organize. This empowerment leads to more sustainable, motivated, and high-performing teams.

	• **Bridge Gaps Between Teams and Leadership:** Agile Coaches often act as translators or facilitators between leadership and delivery teams, ensuring alignment of Agile efforts with business goals and helping leaders adopt servant leadership principles.
2	# Coaches Understand How People Learn ## Learning Models Relevant to Coaching • **Experiential Learning:** People learn best by doing and reflecting on experience. • **Social Learning:** Learning happens through observation, collaboration, and social interaction. • **Cognitive Load Theory:** People can only absorb and process a limited amount of new information at once. • **Motivation Theory:** Intrinsic motivation (interest, autonomy) is stronger than extrinsic (rewards/punishments). ## Coaching Techniques to Support Learning • Encourage hands-on practice with new processes or tools. • Facilitate group discussions and peer learning to share insights. • Break down complex ideas into manageable chunks. • Reinforce learning through repetition and real-world application. • Provide timely, constructive feedback. • Tailor learning approaches to individual and team needs.
3	# Coaches Understand What Makes a Methodology Work ## Beyond Practices and Rules • Methodologies are not just a checklist of practices; they rely on **mindsets, values, and principles**. • The human and social context—trust, communication, collaboration—is key. • Success depends on how teams **adapt** the methodology to their environment. ## Flexibility and Adaptation • Good coaches emphasize **experimentation and continuous improvement**. • Avoid dogmatic adherence; instead, encourage teams to **inspect and adapt** practices. • Understand the organization's culture and tailor interventions accordingly. ## Empowerment and Ownership • Methodologies work when teams feel empowered to own their process. • Coaches help build autonomy and accountability. • Encouraging self-organization fosters sustainable success.

4

The Principles of Coaching

Core Coaching Principles

- **Serve, don't dictate:** The coach supports rather than commands or enforces.
- **Listen actively:** Understand the needs, fears, and aspirations of individuals and teams.
- **Ask powerful questions:** Encourage reflection and self-discovery.
- **Facilitate, don't control:** Guide conversations and decisions without taking over.
- **Focus on strengths and possibilities:** Build confidence and highlight positive behaviors.
- **Promote continuous learning:** Encourage experimentation, feedback, and adaptation.
- **Maintain confidentiality and build trust:** Create a safe space for honest dialogue.

Coaching vs. Managing

- Managers direct and evaluate; coaches empower and enable.
- Coaches work **with** teams, focusing on growth and capability building.
- Coaching is a long-term investment in people and culture.

Coaching Practices

- Use retrospectives and feedback sessions to foster reflection.
- Help teams set achievable goals and celebrate progress.
- Address conflicts constructively.
- Encourage transparency and open communication.
- Support leadership in understanding Agile values and removing impediments.

5

Topic	Key Points to Remember
Why People Resist Change	Change causes fear and uncertainty; resistance is natural.Coaches build trust and use incremental change.
How People Learn	Learning is experiential, social, and cognitive.Coaches support hands-on practice, peer learning, and feedback.
What Makes Methodologies Work	Success depends on mindset, culture, and adaptation.Coaches encourage experimentation and team ownership.
Principles of Coaching	Serve, listen, ask questions, facilitate, focus on strengths.Coaching empowers rather than controls.Builds trust and promotes continuous learning.