Course Transcript

Introducing Agile Software Development

Introduction to Agile Software Development

Learning Objective

After completing this topic, you should be able to

- *identify the principles of agile programming*

1. Waterfall and iterative development

Traditionally, there are two approaches to software development – the waterfall method and the iterative method.

In the waterfall method, development is predictive and occurs as a series of discrete phases, each occurring only once the previous phase has completed.

In the iterative method, development is adaptive and incremental. It occurs as a series of iterations – or cycles, each of which includes all the development phases.

Graphic

The waterfall method's phases are analysis, design, coding, and testing, after which, the software is deployed. The iterative method begins with analysis and architecture. It's phases – planning, design, coding, and testing – are incremental cycles, after which software is deployed.

Using the waterfall method, a team plans software development in fine detail before coding starts. It then completes each development phase in turn to implement the plans.

A project completes with one large release of working software.

Typically, development based on the waterfall model includes four main, sequential phases.

Analysis
During the analysis phase, a team gathers and evaluates requirements, to determine the customer needs that the final software product must meet.

Design
In the design phase, the team translates requirements into a software design and determines how the project will be structured. In a large project, the design process may be broken into two phases – architectural design and then more detailed system design.

Coding

In the coding phase, programmers write, compile, and integrate the program's code based on the finalized design.

Testing

Finally, in the testing phase, the team tests the fully developed software. In larger projects, this typically includes both unit testing of modules and integration testing of the full product.

A significant disadvantage of the waterfall method is that it doesn't enable development teams to respond well to change. Once comprehensive plans are in place, development is expected to follow these plans – making it difficult to adapt to unforeseen changes.

For example, changes in customer requirements – which are common in any project – are difficult and time-consuming to incorporate once development work begins.

Also, misjudgment of the required development time and work during planning can lead to late delivery, cost overruns, or even cancellation of a project well into development.

Mistakes that occur during early planning and design may be carried through a project, ultimately resulting in software failures.

Also, testing occurs after the fact – development teams respond to the results of testing only after a program has been fully developed.

The iterative method of software development overcomes many disadvantages of the waterfall model. It's adaptive because development occurs over several iterations. Each iteration incorporates planning, design, development, and testing, and results in a small release of working software.

The process is incremental because each release builds on the previous one – although each release is also fully functional and designed to meet customer needs.

It's easy to adapt each successive iteration based on what occurred in the last.

Like in the waterfall method, the iterative method includes a set of processes. However, these processes occur in each of multiple iterations.

Four main processes occur in each iteration.

Planning and Analysis

At the start of an iteration, a team plans what functionality to include in a small release, based on a subset of the requirements for a system. For example, it might choose to develop only a section of the required graphical user interface for a program.

Through analysis, the development team elaborates requirements and determines how it can improve development work that has already been done.

Design

An initial design for the software release is revisited and adapted as development proceeds.

Coding

Coding involves creating the required software code, as well as unit and system-level tests. As unit code is completed and tested, it's integrated with other, working code at the system level – which, ideally, is never far from being production-ready.

Testing

Testing occurs at both the unit and over-all system levels, to ensure that developed software meets requirements.

A team continues with iterations until all requirements are met or the deadline for a project is reached.

Advantages of the iterative method are that

there's always a running version of the program

Using the iterative method, you've always got a running version of a program. If you run out of time, for example, you can simply deliver the latest iteration of the program. While it might not have all the functionality the customer required, it's more valuable to the client than a program that doesn't compile.

errors are easy to identify

Because you've got a previous version of a program that you've already tested, and you've made small, known modifications since then, it's easier to identify and fix errors.

errors take less time to fix, and

The iterative method focuses on early and ongoing testing – and errors you detect early in development generally take less time to fix than those you spot later.

early feedback can be obtained

Using the iterative method, feedback on working code can be obtained throughout development, rather than only once development completes.

Question

Match characteristics to the corresponding software development methods. More than one characteristic may match to each method.

Options:

A. Involves rigorous planning before coding work begins
B. Delivers a single large release of software
C. Involves adapting to changes as they occur, during development
D. Delivers software over several, incremental cycles

Targets:

1. Waterfall

2. Iterative

Answer

Using the waterfall method, you plan and design software in fine detail before coding work begins, and deliver this software as a single large release once development and testing are completed.

Using the iterative method, you deliver small, incremental releases of software after each of multiple iterations, adapting each successive iteration based on what occurred in the previous one. Within an iteration, analysis, design, and testing enable ongoing adaptation to changes.

Correct answer(s):

Target 1 = Option A, Option B

Target 2 = Option C, Option D

2. Characteristics of agile development

The concept of agile software development was introduced by a group of developers called the Agile Alliance. This group defined a set of principles for efficient, adaptive software development, in what's known as the Agile Manifesto.

The Agile Manifesto highlights the value of

- individuals and interactions over processes and tools
- working software over comprehensive documentation
- customer collaboration over contract negotiation, and
- responding to change over following a plan

Whereas the waterfall method is predictive and planning-oriented, agile development is adaptive. It focuses on continually responding to change and the learning that occurs as development progresses.

Agile projects are also people-oriented. Whereas the waterfall method standardizes team roles in relation to processes, agile projects adapt processes to capitalize on the unique strengths of individuals and teams.

In practice, agile programming aims to minimize risk and improve efficiency by focusing on the frequent delivery of smaller, incremental releases of working software. At the end of each increment in development, agile teams re-evaluate project priorities.

Within each development period, open communication among team members, collaboration with the customer, testing, and frequent replanning are all ongoing.

Question

Identify the features of agile programming that characterize it as iterative.

Options:

1. It embraces change to help minimize risk
2. It focuses on smaller deliverable units
3. It's predictive and planning-oriented
4. It standardizes people's roles in relation to processes

Answer

Option 1: *Correct. Agile development is iterative because it's adaptive, embraces change, and aims to minimize risk by focusing on the release of smaller deliverable units, or increments.*

Option 2: *Correct. Agile development is iterative because it delivers software in small increments. This helps to minimize risk.*

Option 3: *Incorrect. Agile development is adaptive rather than predictive, and focuses on the delivery of working software rather than on following detailed plans.*

Option 4: *Incorrect. Agile projects are people-oriented and seek to capitalize on the strengths of the individuals in a development team. They involve adapting processes according to the people involved.*

Correct answer(s):

1. It embraces change to help minimize risk
2. It focuses on smaller deliverable units

Agile development depends on four key values.

Communication

Agile development depends on regular, face-to-face communication among all team members so that teams can make decisions and mitigate problems quickly.

For this reason, agile methods are best suited to smaller development teams, ranging from two to about 30 people. Larger teams make effective communication more difficult and cycles longer.

Simplicity

Agile development focuses on simplicity, with teams completing only the work necessary to satisfy existing customer requirements. So teams don't anticipate possible future needs or add software features that aren't specifically required.

This reduces the code that must be built and tested, as well as the need for complex processes and documentation. In turn, this reduces the time to market for software.

Feedback

In agile projects, feedback from customers is obtained frequently, throughout the development process. This ensures work can be adapted early rather than late to meet requirements, and that teams stay focused on these requirements.

Courage

Courage is a key value because agile development depends on open, honest communication, as well as ongoing acceptance of and adaptation to change.

The basic agile values are associated with a set of 12 interdependent principles that inform agile development.

The first six agile principles are

customer satisfaction

The highest priority of an agile development team is to satisfy the customer's existing needs, through frequent delivery of working software.

changing requirements

Agile teams welcome changing requirements during the course of a project. The agile process harnesses change and so helps enhance the customer's competitive advantage.

working software

Each iteration in an agile project results in working software that can be delivered to the customer.

frequent collaboration

During an agile project, developers and their customers collaborate on a daily basis.

motivated individuals, and

Agile projects depend on motivated team members, who can be trusted to fulfill their roles and get their work done independently.

communication

Agile teams prioritize face-to-face communication to ensure that information is conveyed effectively and efficiently.

The remaining six principles are

progress measurement

In agile projects, the primary measurement of success is the working software that teams develop and customer feedback about this.

sustainable development pace

Agile teams are encouraged to maintain a sustainable pace of work – for example, with team members not exceeding 40 hours of work per week. The focus is on consistent, ongoing delivery of software at a pace that can be maintained.

attention to technical excellence

To increase their agility, teams need to focus on providing good design and paying continuous attention to technical excellence.

simplicity

Agile teams streamline the development process by simplifying it – focusing only on what's really needed.

self-organization, and

Agile teams should be self-organizing rather than finely managed. This is in keeping with the need for ongoing collaboration and communication among all team members.

self-reflection

Agile teams reflect on their own performance, continuously identifying and making improvements to the work they complete.

Question Set

Agile methodologies are built on 12 main principles.

Question 1 of 2

Question

Identify six principles of agile software development.

Options:

1. Ensure customer satisfaction
2. Welcome changing requirements
3. Deliver working software frequently
4. Collaborate with business people
5. Build projects around motivated individuals
6. Communicate face-to-face
7. Avoid changing requirements once development work starts
8. Test software rigorously once development work completes

Answer

Option 1: *Correct. An agile team's highest priority is to satisfy customers' existing needs, by delivering working software frequently.*

Option 2: *Correct. Agile development is adaptive – change is welcomed and teams adapt to it as development proceeds.*

Option 3: *Correct. Agile teams focus on delivering working software after each iteration, or short development period.*

Option 4: *Correct. Agile developers should collaborate regularly with customers as development proceeds.*

Option 5: *Correct. Agile projects should be built around motivated development team members, who can be trusted to make decisions and get the required work done independently.*

Option 6: *Correct. In an agile team, face-to-face communication is necessary for effective collaboration, among team members and with the customer.*

Option 7: *Incorrect. Agile teams seek to identify and adapt to changing requirements throughout a project.*

Option 8: *Incorrect. Agile teams focus on continuous testing, throughout the development process. The later problems are identified, the costlier they are to fix.*

Correct answer(s):

1. Ensure customer satisfaction
2. Welcome changing requirements
3. Deliver working software frequently
4. Collaborate with business people
5. Build projects around motivated individuals
6. Communicate face-to-face

Question 2 of 2

Question

Which are remaining principles of agile software development?

Options:

1. Regularly measure success based on working software
2. Maintain a sustainable development pace

3. Focus on technical excellence
4. Simplify development
5. Use self-organizing development teams
6. Continuously reflect on performance
7. Predict and respond to customers' likely future needs
8. Deliver software in a single large release

Answer

Option 1: *Correct. Agile teams use the working software they develop in each iteration – and customer feedback about this – as the primary method of measuring success.*

Option 2: *Correct. Agile team members are encouraged to work at a sustainable, consistent pace.*

Option 3: *Correct. Agile teams should seek to enhance their agility by providing good design and maintaining a focus on technical excellence.*

Option 4: *Correct. Agile teams simplify development by focusing only on work that's really needed to meet customer requirements.*

Option 5: *Correct. Agile teams should be independent and self-organizing, so they can collaborate and adapt to change easily.*

Option 6: *Correct. Agile teams continuously assess and improve on completed work.*

Option 7: *Incorrect. Agile methodologies are adaptive rather than predictive. The focus is on meeting existing customer needs as they arise, rather than on predicting possible future needs.*

Option 8: *Incorrect. Agile development is iterative and incremental, with a focus on frequent delivery of small releases.*

Correct answer(s):

1. Regularly measure success based on working software
2. Maintain a sustainable development pace
3. Focus on technical excellence
4. Simplify development
5. Use self-organizing development teams
6. Continuously reflect on performance

Many agile development methodologies are available, and each interprets the agile principles slightly differently. For example, some methodologies place more emphasis on processes and documentation than others.

3. Summary

The waterfall method of software development is predictive and involves analysis, design, coding, and testing as sequential phases. In contrast, the iterative method focuses on developing software over multiple cycles, each resulting in an incremental but working release.

Agile software development is based on the iterative method. It focuses on streamlining development by making it highly adaptive, with continuous collaboration, testing, and feedback incorporated in the development process.

Extreme Programming (XP) and Scrum

Learning Objectives

After completing this topic, you should be able to

- *identify the main principles of XP*
- *identify the main tenets of Scrum*

1. Extreme Programming

Extreme Programming – also known as XP – is one of the best-known agile methodologies. It provides a programmer-centric model that's focused on the ongoing, rapid delivery of small releases of software.

With XP, the interval between software releases is usually 30 to 180 days. This period is further broken down into iterations, each lasting between one and four weeks.

Each iteration results in production-ready code – working code that has been fully tested and can be integrated in a release.

To achieve the required level of efficiency, all members of a small, self-directed team of programmers collaborate closely with one another and with an on-site customer.

The team sets its own coding standards and handles the planning of coding tasks based on a set of customer user stories – each of which describes a single functionality the customer requires of the software product.

The focus of design is on simplicity, and ongoing testing, customer feedback, and refactoring are integrated in the development process.

Note

Refactoring refers to the re-design of a program or of code that has already been implemented, typically to improve its performance.

XP is associated with four values – communication, simplicity, feedback, and courage. These overarching values must guide an XP team if it's to succeed in meeting customer needs quickly, efficiently, and with minimal external direction.

At a more concrete level, XP is associated with a specific set of 13 principles.

Planning game

In the XP model, what's referred to as the *planning game* is the main planning process that simplifies both release and iteration planning.

Small releases

An emphasis in XP is on releasing software products as quickly as possible, to reduce time to market and enable timely feedback.

System metaphor

An XP project team should keep a simple statement about what a system is meant to do and how it's meant to do it. This statement should take the form of a metaphor, which clarifies what's required using easily recognizable objects or familiar concepts. For example, a shopping cart is a popular metaphor for the functionality that enables online shoppers to select items to purchase and to add these to a repository. The metaphor should be sufficiently broad that it won't change even as user stories – which identify requirements in more detail – are modified. It ensures team members stay focused on a clear, high-level objective.

Simple design

A core XP principle is simplicity – always use the simplest possible design to provide the functionality described in a user story, and don't include functionality that isn't specifically required.

This principle is based on the need for rapid releases of useful software. Business needs can change quickly, so it's critical to focus on meeting existing needs efficiently, and not to waste time on developing functions that may never be used.

Ongoing testing

Ongoing testing that's fully integrated in the development process is central to the XP model. Programmers write unit tests even before they start writing code. This informs and improves the design and development processes. Testing then continues throughout development, and should be automated where possible. This means that once an iteration completes, code has been fully tested and is ready for integration in a release.

Refactor

Rather than designing a complete system and then developing it, an XP team refactors working code on an ongoing basis. In other words, the team modifies system design progressively and when needed, as more is learned. For example, refactoring may involve completely changing the structure of a class – or even of a full subsystem – that has already been written and tested, to improve overall system design.

Pair programming

The XP model encourages pair programming, in which two or even three programmers work at a single workstation. This enables ongoing discussion and resolution of design, testing, and programming concerns, which can result in better quality code.

During release planning, the programming team gathers customer requirements in the form of user stories. It assigns each story a number of points, based on the estimated time and work required to implement it. The team also determines the total number of story points it can include in a release, given available resources and a specified release date.

The customer then prioritizes the user stories, selecting a set that doesn't exceed the maximum number of story points. If the resulting set is judged by the customer to provide business value, the project proceeds.

During iteration planning, the team translates a subset of the selected user stories into corresponding coding and testing tasks. Members of the team then assign these tasks among themselves.

The XP principles determine how programming occurs.

Collective code ownership

Any member of an XP team is entitled to alter any system code, at any time. So although each programmer focuses on particular coding tasks, everyone on the team shares ownership and responsibility for the full system code.

Continuous integration of code

The XP model relies on constant rebuilding and testing of the full code base for a system. So although programmers continually add new working code they've developed, the full code base is never far from being in a production-ready state.

Sustainable pace

XP team members are encouraged not to work more than 40 hours per week. Instead the emphasis is on maintaining a sustainable pace, with delivery that's consistent and reliable.

Shared workspace

XP team members are encouraged to learn about all relevant technologies and to be familiar with all system requirements, rather than focusing only on specialized areas of development. Also, all members of the team typically work in the same room. This encourages full collaboration and a sense of shared ownership.

Coding standards

An XP team is responsible for defining and implementing a consistent set of coding standards. This helps ensure that everyone on the team writes code that can be easily understood by other team members.

On-site customers

Constant, direct access to a customer – who sits in the same workspace as programmers and acts as a team member – enables fast software development. Requirements can be checked or modified, and approval can be obtained immediately.

All the XP practices support one another. Although many agile teams implement some of these practices, adopting them in full requires a high degree of discipline, teamwork, and skill.

Graphic

Description of diagram showing how the XP practices support one another:
The shared workspace of programmers and the on-site customer support the planning game and pair programming. Pair programming supports refactoring,

ongoing testing, and continuous integration of code. Continuous integration of code supports collective code ownership and coding standards. The planning game supports system metaphors, simple design, and small releases, which in turn enable a sustainable pace of development.
Description ends.

Question

Which are key XP principles?

Options:

1. Collective code ownership
2. Thorough investigation of all potential user requirements prior to initial software planning
3. Pair programming, with two or more programmers sharing each workstation
4. Continuous code integration
5. Comprehensive post-production testing

Answer

Option 1: *Correct. One of the main XP practices is collective code ownership. Every member of an XP programming team has full access to all the code for a release and can make any necessary changes to it.*

Option 2: *Incorrect. The focus in XP is on rapidly delivering small releases of software that meet specific, current needs, rather than on delivering software that attempts to meet a wide range of anticipated or potential needs.*

Option 3: *Correct. The XP model encourages pair programming because this enables real-time discussion and collaboration between programmers. Concerns can be addressed as they arise, resulting in better-quality code.*

Option 4: *Correct. Continuous integration is one of the core XP practices. During a development iteration, members of an XP programming team continually add to, rebuild, and retest a working code base.*

Option 5: *Incorrect. XP emphasizes ongoing testing. Unit tests are developed before code is written, and testing and refactoring continue as part of the development process. This makes comprehensive post-production testing unnecessary.*

Correct answer(s):

1. Collective code ownership
3. Pair programming, with two or more programmers sharing each workstation
4. Continuous code integration

Because one focus of XP is the collaboration and equal participation of all team members, an XP team may typically include only a few role designations – customer, development coach, business coach, developer, and tracker. However, it can include all of the listed roles.

Customer
The customer creates and prioritizes user stories, as well as providing ongoing feedback to the team.

Development coach
The development coach monitors the development process to identify potential problems, and guides or mentors team members on the use of XP techniques.

Business coach
A business coach can be used to represent and guide the customers. They will have a good understanding of the business requirements of the project and of XP. They will help the customers to write user stories and acceptance tests.

Developer
Developers are the programmers who plan, write, and test code based on the customer's user stories.

Business Analyst/Systems Engineer
These act as subject matter experts for XP development teams. They can assist customers with technical information and represent customers' business needs if required.

Tracker
The tracker monitors the team's progress to ensure it stays on schedule, calling for adjustments or a redistribution of workloads if necessary.

Tester
Specific testers can be used to develop the test plan and work with customers on acceptance tests. They need to ensure that user stories are testable and they run acceptance tests.

Big Boss
The XP Big Boss is responsible for assembling the XP development team, and looking after workspace and equipment requirements. They will represent the team, when required, to those outside the XP team.

Question

Match roles in an XP team to their key functions.

Options:

A. Customer
B. Tracker
C. Developer
D. Development coach

Targets:

1. Specifies the functionality software must provide
2. Monitors the team's progress
3. Implements and tests code
4. Mentors team members on applying XP techniques

Answer

The customer supplies the requirements software must meet in the form of a prioritized set of user stories.

A tracker monitors the team's progress to ensure it stays on schedule, calling for adjustments if necessary.

A developer plans, writes, and tests code to implement the customer's requirements.

A development coach guides team members through the XP development process.

Correct answer(s):

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

2. Scrum

Whereas XP focuses on programming practices, the Scrum methodology focuses on agile project management. It's often used together with XP or with other agile methodologies.

Scrum takes its name from the scrum in rugby, in which players form a tight knot and together execute a plan for passing the ball in a desired direction.

The Scrum methodology is based on short daily meetings – known as scrums – in which all team members collaborate on managing a development project.

Note

Each scrum meeting should last between 10 and 15 minutes.

In Scrum meetings, each team member answers three questions:

- What work have I completed since the last scrum, and why?
- What do I plan on completing between now and the next scrum?
- Do I have any roadblocks or problems that the team can help overcome?

Also borrowed from rugby is the concept of a *sprint*. In Scrum, each sprint (iteration) is a fixed-length development period with a clear goal, consisting of an agreed set of work items to implement.

Each successive sprint builds on the last, and planning occurs between the sprints.

Note

Typically, one sprint lasts from two weeks to a month.

A Scrum team uses three types of backlogs to plan and manage a development project:

a sprint backlog
A sprint backlog details the work items that team members have agreed to complete in a given 30-day sprint. Once a sprint starts, no changes should be allowed to interfere with its goal.

a release backlog, and
A release backlog details all the work items that must be completed before a software product can be released.

a product backlog
A product backlog details all changes and ideas for implementation that don't directly support the agreed goal for the current sprint. This backlog then informs the planning for subsequent sprints.

To track its progress in completing the agreed work items during a sprint, a Scrum team uses what's known as a *burndown chart*.

In the chart, the straight line indicates the ideal workflow, in which work items are completed in a completely steady, uniform manner.

A second line, which is updated daily, represents the team's actual progress. This makes it clear when adjustments are needed to bring development back on schedule.

Scrum emphasizes a management process based on empirical control, with planning that's iterative and incremental.

In addition, a key principle is that a Scrum team must be self-organizing. Team members organize and manage their own work, based on the goals for a sprint and on daily scrum meetings.

Core values associated with the success of the team include commitment, focus, openness, respect, and courage.

The ideal Scrum team includes four to nine members.

However, the methodology is scalable. If multiple teams are involved in a project, each team's daily scrum meeting can be followed by a second scrum. In the second scrum, a representative from each team collaborates to ensure that all the teams' efforts are coordinated.

A focus in Scrum is on the equal participation of all team members. Accordingly, apart from the team itself, only two Scrum roles are specifically defined:

Product Owner and
The Product Owner represents the interests of the customer or end user. This person is also responsible for liaising between the team and the customer.

Scrum Master
The Scrum Master facilitates and reports on the work of the team, focuses the team on the highest-priority tasks, and assists in overcoming any roadblocks. Certification training at three levels is available for Scrum Masters.

Question

Which are main tenets of the Scrum methodology?

Options:

1. Project planning is iterative and incremental
2. Development work is timeboxed and sprint-driven
3. Project team members are self-organizing
4. A trained Scrum Master assumes full responsibility for assigning tasks and managing workflow
5. Programmers work in pairs

Answer

***Option 1:** Correct. A Scrum team plans the work items to complete in each fixed-length sprint. Planning occurs again before the next sprint, and successive sprints build on one another.*

***Option 2:** Correct. In a Scrum team, particular work items are assigned to each of a series of fixed-length sprints. Once a sprint begins, the team focuses only on the goal of that sprint.*

***Option 3:** Correct. All members of a Scrum team organize and manage their own work, based on the goals for a sprint and on daily scrum meetings.*

***Option 4:** Incorrect. The Scrum Master focuses team members on high-priority tasks and facilitates their work – but all team members participate in both managing and completing the required work.*

***Option 5:** Incorrect. Pair programming is a concept associated with XP. Scrum focuses on project management, rather than on programming practices. However, XP principles such as pair programming may be combined with an implementation of Scrum.*

Correct answer(s):

1. Project planning is iterative and incremental
2. Development work is timeboxed and sprint-driven
3. Project team members are self-organizing

Question

Match each role on a Scrum team to the corresponding description.

Options:

A. Product Owner
B. Scrum Master
C. General team member

Targets:

1. Represents the customer's interests
2. Guides and facilitates the development process
3. Participates in managing and completing the work designated for each sprint

Answer

The Product Owner represents the interests of the customer or end user of the software being developed.

A Scrum Master focuses the efforts of the team, assists in overcoming roadblocks, and reports on team progress.

All team members participate in planning and completing the work items designated for each 30-day sprint.

Correct answer(s):

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

3. Summary

Extreme programming, or XP, is a programmer-centric methodology that focuses on the ongoing, rapid delivery of small releases of software. Testing, refactoring, and responding to customer feedback are all fully integrated in the development process, and very close collaboration among team members is required.

Scrum is a methodology for managing agile development projects. It involves iterative, incremental planning, which occurs between spurts of development work driven by clear goals. Team members manage their own work and track progress through daily meetings.

Lean Software Development and Kanban

Learning Objectives

After completing this topic, you should be able to

- *recognize the seven principles of lean software development*
- *identify the main features of the Kanban approach to agile software development*

1. Lean software development

Lean software development employs a set of principles derived from the lean production processes pioneered by manufacturing companies like Toyota.

Unlike other agile methodologies, it doesn't prescribe the use of specific development methods. Instead, it provides guidelines for streamlining the development process so that teams can meet customer needs much more efficiently.

Like Scrum, which focuses on project management, lean principles can be incorporated into other methodologies.

Lean software development is informed by seven key principles:

eliminate waste

The principle of eliminating waste is at the core of any lean system. In the context of software development, waste includes time and effort that doesn't directly add value to the end product. Examples are unnecessary documentation and software features, work on design or code that's never implemented, and waiting for others to complete tasks before proceeding with new work.

To eliminate waste, you need to start by analyzing each step in a team's development process. What waste does this step involve, and what value – if any – does it add? Unnecessary steps can then be removed.

promote and incorporate continuous learning

Efficient software development depends on getting things right early on, instead of fixing problems later. In turn, this requires a focus on building learning into the development process.

For example, the process should enable early and ongoing testing, and early customer feedback. It should be broken into time-boxed iterations so that learning from one iteration can inform the next. Also, team members should collaborate and learn from one another.

delay decisions until they're needed

Making decisions about software design and code as late as possible reduces the risk that new information – or a change in business needs – will invalidate those decisions.

deliver software as fast as possible

An emphasis in lean development is on reducing time to market, to meet customers' needs faster and more efficiently.

This involves planning smaller releases of software after shorter periods, focusing on meeting customers' existing – rather than anticipated – needs, and identifying and removing bottlenecks from the development process.

empower the programming team

A lean system can't be imposed only from above – those closest to the work need to support and apply the lean principles. So it's vital that all members of a programming team are authorized and motivated to do this.

For example, all team members should have the authority to implement decisions that will improve either a process or the end product. They should also participate in monitoring progress, and have access to all required resources.

build a focus on system integrity into the development process, and

Software has integrity if it's well-designed, meets the customer's needs, and will continue to be useful. Incorporating a focus on integrity in the development process results in better efficiency, as well as higher-quality software.

To ensure integrity, system testing and customer feedback should occur throughout the development process, instead of only after the process completes. Programmers should also be encouraged to refactor or completely redesign code when this will improve its performance.

focus on the whole system

To implement a lean development process, all team members must be able to focus on a whole system and how its parts will integrate, rather than only on specialized system components.

This requires collaboration and an emphasis on the breadth as well as the depth of team members' knowledge. It also requires ongoing testing at the system level, instead of only at the unit or component level.

Question

Lean software development is associated with seven main principles.

Which are among these principles?

Options:

1. Make decisions as late as possible, to prevent new information from invalidating them
2. Streamline the development process by identifying and removing steps that don't add value
3. Gear development toward fast, small releases of software

4. Incorporate learning into the development process
5. Focus on exceeding customer requirements
6. Encourage specialization by development team members

Answer

Option 1: *Correct. Delaying decisions until they have to be made helps reduce the risk that new information or a change in business needs will invalidate the decisions.*

Option 2: *Correct. Implementing a lean process involves eliminating waste. In the context of software development, this includes eliminating activities that don't add direct value to the end product and that aren't strictly required.*

Option 3: *Correct. A key focus in lean software development is reducing time to market.*

Option 4: *Correct. Continuous learning through ongoing testing and customer feedback is incorporated in a lean development process to improve both quality and efficiency. For example, testing and the consequent reworking of code occurs throughout, rather than only after the process completes.*

Option 5: *Incorrect. A lean development process focuses on meeting customers' existing needs as quickly and efficiently as possible, rather than on exceeding requirements.*

Option 6: *Incorrect. In a lean system, breadth of knowledge is emphasized. All team members should focus on how components will integrate and work together in a full system, rather than looking at specialized components in isolation.*

Correct answer(s):

1. Make decisions as late as possible, to prevent new information from invalidating them
2. Streamline the development process by identifying and removing steps that don't add value
3. Gear development toward fast, small releases of software
4. Incorporate learning into the development process

Question

Which are remaining principles for lean software development?

Options:

1. Empower a development team to direct its own work

2. Incorporate a focus on system integrity
3. View components in terms of how they'll integrate and work together in a full system
4. Ensure all members of a development team meet on a daily basis
5. Minimize changes to working code

Answer

Option 1: *Correct. In a lean system, all team members should be authorized to apply lean principles in practice – for example, by altering code to improve its design.*

Option 2: *Correct. Focusing on system integrity during development prevents the need for later changes, as well as improving software quality.*

Option 3: *Correct. In a lean system, all team members should be encouraged to view their work on specific components in terms of how all components will integrate and work together in a whole system.*

Option 4: *Incorrect. Lean software development doesn't prescribe specific methods – like daily meetings – for managing a development project. Agile methodologies such as Scrum do this.*

Option 5: *Incorrect. A lean team learns as the development process progresses, testing and refactoring code as necessary to improve its performance.*

Correct answer(s):

1. Empower a development team to direct its own work
2. Incorporate a focus on system integrity
3. View components in terms of how they'll integrate and work together in a full system

2. Kanban

Kanban evolved from lean principles and, like them, has its roots in the manufacturing processes of companies like Toyota.

The term "Kanban" is a Japanese one meaning card or board. In the Toyota Production System – known as TPS – physical cards are used at different points in an assembly line, to control the flow of work and materials.

For example, a particular component is assembled at station A and painted at station B. When the employee at station B has only three components left to paint, this person carries a card to station A, stating that 10 more components should be delivered for painting.

Someone further along the line will deliver a card to station B as soon as a new batch of painted components should be delivered for fitting to vehicles.

This ensures that materials and work are "pulled" rather than "pushed" through the assembly line. No bottlenecks occur because workflow is controlled based on the capacity of each successive point in the system.

The Kanban methodology involves using a presentation board divided into columns to manage the flow of work items through a process, so that workflow doesn't exceed capacity at any one point.

In the context of software development, all user stories for inclusion in an iteration are in column A. Each user story should be a required functionality – referred to as a *minimal marketable feature*, or MMF.

Note

It's common to limit the total number of user stories to half the number of members on a development team. For example, a team with 30 members might include 15 user stories.

Further columns represent successive steps in the development process. For example, these might include acceptance, development, testing, and deployment.

Each column representing a step is divided into two parts:

work in progress and
When work in a particular step starts on a work item, the item is moved to the top section of the column for that step.

a buffer
When a step for a work item is completed, the work item moves into the buffer area for that step. It then waits there, to be "pulled" to the next step by a developer who's ready to perform it.

Each step is assigned a maximum capacity – for example, two work items. This value will depend on how many developers are available to work on the step.

Work items move from left to right on the presentation board as they pass through each step in the process – but only once there's enough capacity at the next step.

Characteristics of a Kanban development system are that it

streamlines workflow
Using a Kanban system ensures that work in progress – known as WIP – is always limited according to workflow status. So time and effort aren't wasted in continuing to produce output from earlier steps when doing this would cause bottlenecks further down the line.

relies on self-organizing teams

Kanban relies on developers to "pull" work items through the process, rather than on assigning tasks with fixed deadlines to each developer before work starts.

involves incremental development

The Kanban system assumes an incremental approach to software development, with each step building on work completed in a previous step.

makes it easy to measure lead times and track progress using empirical data, and

A Kanban-style presentation board makes it easy for a team to track its progress at any point. It also enables empirical measurement of lead times – based on the average time it really takes to develop and deploy each MMF, rather than on guesswork about the time needed.

enables continuous optimization of release plans

Kanban maximizes the overall throughput of a development system by continuously ensuring that work moves in accordance with real capacity.

Also, the use of a presentation board makes it easy for team members to distribute their efforts as efficiently as possible to meet release dates. For example, a backlog of work items in the buffer for a particular step can prompt developers with less work in progress to assist with that step.

A Kanban system doesn't prescribe team roles. It also doesn't specify the ideal length of an iteration or the steps it must include.

However, Kanban can easily be merged with other more prescriptive methodologies like Scrum – which uses sprints rather than workflow status to limit the amount of work in progress.

Note

During a sprint of fixed length, a Scrum team focuses only on an agreed set of work items. All other work items are placed in buffers until the sprint's goal is met.

For example, a Scrum team may find that a fixed-length sprint prevents it from responding effectively to defects as they arise, in software that has already been deployed. It may then use Kanban to prioritize these issues, while still addressing the work items included in the current sprint.

Question

Which are among the key features of the Kanban approach to software development?

Options:

1. The amount of work in progress is always limited by real capacity
2. A presentation board provides a visual representation of workflow status at all times
3. Lead times can be measured empirically
4. Development teams include clearly defined roles
5. A team focuses only on an agreed set of work items for a fixed period

Answer

Option 1: *Correct. A Kanban system ensures that the flow of work items through a development process doesn't exceed capacity at any point, by constantly tracking workflow status.*

Option 2: *Correct. A Kanban board provides a visual representation of all the work items in progress and of all those awaiting attention at any one time, across the development process.*

Option 3: *Correct. A Kanban system enables empirical measurement of lead times, based on how long it really takes work items to move through the development process.*

Option 4: *Incorrect. Kanban doesn't identify or prescribe specific team roles.*

Option 5: *Incorrect. Scrum, rather than Kanban, uses the concept of fixed-length sprints, during which a team focuses only on an agreed set of work items.*

Correct answer(s):

1. The amount of work in progress is always limited by real capacity
2. A presentation board provides a visual representation of workflow status at all times
3. Lead times can be measured empirically

Question

Identify additional features of Kanban.

Options:

1. The development process is incremental
2. Development teams are self-organizing
3. Release plans are continuously optimized
4. Each iteration lasts from 30 to 180 days

5. For all features included, each step in the development process must complete before work on the next step begins

Answer

Option 1: *Correct. In a Kanban approach, each development step and iteration builds on previous ones.*

Option 2: *Correct. Kanban relies on all members of a team to "pull" work items through the development process, rather than on these items being pushed out to them.*

Option 3: *Correct. Kanban maximizes the overall throughput of a development system by continuously ensuring that work moves in accordance with real capacity.*

Option 4: *Incorrect. Unlike more prescriptive agile methodologies, Kanban doesn't specify an ideal duration for iterations.*

Option 5: *Incorrect. This describes a traditional or "waterfall" approach to development. With Kanban, work on different steps may be in progress at any one time.*

Correct answer(s):

1. The development process is incremental
2. Development teams are self-organizing
3. Release plans are continuously optimized

3. Summary

Lean software development is informed by a set of principles for streamlining the development process and improving its efficiency.

Kanban incorporates the lean principles and focuses on optimizing the flow of work through a development system. It does this by constantly limiting work in progress based on the real capacity of each point in the system.

FDD and ASD

Learning Objectives

After completing this topic, you should be able to

- *identify the features of FDD*
- *recognize what happens at the three phases in an ASD project*

1. FDD

Unlike most other agile methodologies, which outline sets of principles or best practices, Feature Driven Development – or FDD – provides a prescriptive model for agile development.

The key characteristic of FDD is that all aspects of the software development process are planned, managed, and tracked at the level of individual software features.

FDD specifies five processes for teams to complete in order:

1. develop an overall model
2. build a features list
3. plan by feature
4. design by feature, and
5. build by feature

The first step in FDD is gathering expert knowledge about what a system must do, and integrating this into a unified domain model, which is a conceptual model of the system that must be built.

All subsequent steps are driven by this domain model. Once it has been developed, it's broken down into its component features. Planning, design, and building of these features then proceeds.

For management purposes, FDD organizes system requirements at three levels.

Subject areas
Subject areas typically correspond to general business practices. An example of a subject area could be "Managing inventory."

Feature sets
Subject areas are broken down into feature sets, each of which typically represents a business activity. Examples of feature sets associated with a subject area like "Managing inventory" could be "Tracking stock levels", "Ordering stock", and "Tracking purchase orders."

Features
Each feature set is broken down into a list of specific features. A feature is a small function that will add value for the customer, and is expressed as an action that causes

a result to an object.

In the example "Calculate the cost of outstanding purchase orders," "calculate" is the action, "cost" is the result, and "outstanding purchase orders" is the object.

Each feature is then translated into a list of programming tasks, which can be estimated and assigned to team members.

Subject areas and feature sets are used only for high-level planning and tracking; all concrete management of the development process is based on individual features.

A team works on a small set of assigned features for a period of fixed length, typically not exceeding two weeks. It then moves onto a new set of features.

Multiple teams may work simultaneously on different sets of features.

Agile methodologies like Extreme Programming – or XP – and Scrum focus on team collaboration and concepts like collective code ownership.

In contrast, FDD relies on team members assuming specific, discrete roles.

The roles defined on an FDD team are

Project Manager
A Project Manager is responsible for administering all aspects of a project, including its budget, and for ensuring appropriate reporting of progress.

Chief Architect
A Chief Architect is responsible for the overall design of a system. This person runs all design sessions and code reviews, and makes final technology decisions.

Development Manager
A Development Manager is responsible for coordinating the daily activities of the development team and for managing resource issues.

Chief Programmer
A Chief Programmer is a senior developer who's assigned responsibility for one or more feature sets. This person oversees and participates in design and development for the relevant feature sets.

Class Owner
A Class Owner is a developer who works under the direction of a Chief Programmer to design, code, test, and document features.

Domain Expert
Also sometimes called a Subject Matter Expert or SME, a Domain Expert is any stakeholder with expert knowledge about the business requirements software must meet. This can include anyone with expert knowledge about a business and its needs, such as end users, clients, and business analysts.

Tester

A Tester is responsible for testing each feature to verify that it performs as required.

Deployer, and

A Deployer is responsible for the deployment of code to different environments, and for the definition or conversion of data into the required formats.

Technical Writer

A Technical Writer is responsible for creating and maintaining all documentation that end users require for the developed software.

FDD teams use several specific tools to track and report progress. These include

a task list for each feature

A task list identifies all the tasks that must be completed to implement a particular feature, which can then be used to track progress in developing the feature.

a milestone table for each feature

For each feature, a team creates a table for recording the date at which each of the six milestones is reached. These milestones are labeled Domain Walkthrough, Design, Design Inspection, Code, Code Inspection, and Promote to Build.

The table enables the team to track where each feature is in its development cycle.

a milestone table for each feature set

The milestones recorded for each feature in a particular feature set are included in a separate table, to enable tracking at the level of feature sets.

a line graph showing completed features across a project, and

A line graph shows the cumulative total, by day or week, of all features that have been completed across a project.

a feature set progress report

A feature set progress report tracks progress on all the feature sets in a project, using color coding to distinguish those that are on track, behind schedule, or completed.

For each feature set, the report details the number of features included, the percentage of work completed, the targeted completion date, and the name of the Chief Programmer assigned to that set.

Question

Which are characteristics of FDD?

Options:

1. A domain model determines the software features a team identifies, plans, designs, and builds

2. All aspects of the development process are managed in relation to individual system features
3. An emphasis is on team collaboration, with collective code ownership
4. System requirements evolve as development progresses

Answer

Option 1: *Correct. The first step in the FDD process is building a domain model that integrates business requirements for a system. Based on this model, a team then builds a feature list and plans, designs, and builds the required features.*

Option 2: *Correct. Development work is assigned, managed, and tracked at a per-feature level.*

Option 3: *Incorrect. FDD prescribes nine distinct roles for members of a development team. Unlike methodologies such as XP and Scrum, it doesn't emphasize the sharing of responsibilities by all team members.*

Option 4: *Incorrect. With FDD, a domain model is created at the outset of a project to incorporate system requirements.*

Correct answer(s):

1. A domain model determines the software features a team identifies, plans, designs, and builds
2. All aspects of the development process are managed in relation to individual system features

2. ASD

The concept of Adaptive Software Development – or ASD – was developed by Jim Highsmith, based on complex adaptive systems theory.

ASD moves away from the idea of development as a mechanistic process to one that's organic and highly responsive.

It involves viewing the software development process as a collaborative learning cycle, with three key phases – Speculate, Collaborate, and Learn. These terms replace the traditional ones of "plan", "build", and "review."

With ASD, a development team is seen as an adaptive system that consists of

- agents, including team members and other stakeholders
- environments, including those that are organizational, technological, and process-based, and
- emergent outcomes – or results

There's no hierarchy on an ASD team. All members and stakeholders are viewed as independent agents. They interact with and learn from one another, together influencing the way a software system evolves.

Rather than directing the work of others, a Project Manager facilitates an ASD project, setting its direction and providing the environment in which the agents can collaborate.

In a traditional – or mechanistic – development model, expected inputs are processed with the aim of producing expected results.

In the ASD model, inputs can be expected or unexpected, the system follows a pattern rather than a process, and results include those that simply emerge, as well as those that are expected.

In the ASD lifecycle, development is incremental and iterative. It occurs through repeated cycles in which team members move from speculation to collaboration, to learning.

The ASD lifecycle includes five steps.

Project initiation

Project initiation involves speculation. At the outset of a project, all team members and stakeholders – including the customer – participate in a workshop. They establish a project's mission, objectives, constraints, and key risks. They then identify system requirements and estimate product size and scope. This lays the groundwork for the remaining steps.

Adaptive cycle planning

Adaptive cycle planning is the first step in a loop that iterates throughout a project. It involves planning – or speculating about – the number and length of required cycles per development period, writing an objective statement for each cycle, and assigning technology and components to the cycles. It also involves developing a project task list.

In each successive project cycle, planning from the previous cycle is revised as necessary, based on what the team has learned.

Concurrent component engineering

Concurrent component engineering involves collaboration; it's the phase during which the team collaborates to implement the planned work. Team members or subteams generally work on components simultaneously and integrate their work products.

The functionality actually built will depend on what's achieved during the time assigned to each cycle and to the total project. Compromises on the planned functionality may be made if time runs short – or extra functionality may be added if it takes less time than expected to meet the planned objectives.

Quality review

Each development cycle ends with learning, obtained through quality reviews.

It's recommended that each quality review include feedback from end users who've tried using the developed software. A quality review should also include software inspections, and end with a postmortem. In a postmortem, the team evaluates its progress and the effectiveness of the processes it has been using thus far. It also brainstorms ideas for resolving any identified problems.

The team then revisits the adaptive cycle planning phase, revising earlier plans based on what it has learned in preparation for the next cycle.

Final QA and release

The final QA and release phase occurs once at the end of a project, when the final product and all required information is handed to the customer for approval and then release.

Key characteristics of ASD that make it agile are that it's

- iterative, with planning, development, and reviews incorporated in each of multiple cycles
- time-boxed, with cycles allocated fixed durations
- mission-driven, with each cycle planned based on a project mission
- component rather than task-based, and
- designed to accommodate changes in each cycle

Question

Match each phase in the ASD lifecycle to a description of what it involves.

Options:

A. Project initiation
B. Adaptive cycle planning
C. Concurrent component engineering
D. Quality review
E. Final QA and release

Targets:

1. All team members and stakeholders plan a project's high-level objectives
2. Team members plan fixed-length development cycles and revisit these plans
3. Team members collaborate to complete planned work, adjusting what's done based on time restrictions
4. Team members learn from the results of a cycle and feed this information back into planning for the next cycle
5. The team hands the developed product to the customer

Answer

To initiate a project, team members and stakeholders participate in a workshop. They establish the project's mission, objectives, constraints, scope, and key risks. This involves speculation.

Adaptive cycle planning occurs before each cycle and is informed by learning from the previous cycle.

In the concurrent component engineering phase, team members interact to complete the planned work for a cycle.

A quality review ends each cycle. Learning from the results then informs planning for the next cycle.

A project ends with a final QA and release phase, during which the final product is handed to the customer for approval and then release.

Correct answer(s):

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

Target 5 = Option E

3. Summary

FDD provides a feature-based model for agile software development. Based on a conceptual domain model of what a system must do, required features are identified, planned, designed, and built. Team members have prescribed roles, and progress is tracked at the level of completed features and feature sets.

ASD is a more organic approach, in which all team members interact as peers. The development process includes multiple cycles in which teams move from speculating, to collaborating, to learning.

AUP and EssUP

Learning Objectives

After completing this topic, you should be able to

- *identify what occurs at each stage of an AUP project*
- *identify the main features of EssUP*

1. AUP

The Unified Process – or UP – is a detailed framework for the iterative and incremental development of software.

The Agile Unified Process – known as AUP – simplifies UP, adapting it specifically for agile software development.

In AUP, a project proceeds through four serial phases in turn.

Inception
In the Inception phase, the team establishes the initial scope of a project and a potential architecture for the system to be developed. It also secures initial project funding and stakeholder acceptance.

Elaboration
In the Elaboration phase, the goal is to improve the team's understanding of system requirements and to validate the selected architecture for the system.

Construction
In the Construction phase, working software is developed on an incremental basis to meet the highest priority needs of the customer.

Transition
In the Transition phase, system testing is completed and software is deployed.

In addition to stages, AUP identifies seven specific disciplines:

- Modeling – creating a conceptual model of the required software
- Implementation – building the software
- Testing – verifying and validating the software
- Deployment – implementing the software in a production environment
- Configuration Management – managing changes and implementing version control
- Project Management – keeping a project on schedule and within budget, and ensuring it meets requirements, and
- Environment – ensuring resources and processes are in place to facilitate development

Note

These disciplines are derived from the set of nine disciplines defined in the UP.

Each of the four phases in AUP is divided into one or more iterations – and during a single iteration, it's typical that all seven disciplines will be practiced.

As with other agile methodologies, the focus of each iteration is on developing a small, production-worthy piece of software.

Each iteration builds on the last to result in the full, required software system – so development is incremental as well as iterative.

Graphic

A chart illustrates how each of the seven disciplines is used – to varying degrees – within each of the four phases, including Inception, Elaboration, Construction, and Transition. For example, the most modeling occurs during the Inception and Elaboration phases, although some modeling also occurs in the other two phases. The most implementation occurs in the Elaboration and Construction phases.

The chart also shows that each phase is divided into one or more iterations.

AUP emphasizes early development of a workable system architecture. This is conceptualized in the first phase – at project inception – and then undergoes elaboration.

A further emphasis is on prioritizing high-risk elements of a project and developing those early on. Typically, a list of prioritized risks is created and maintained throughout the development process.

The main principles of AUP are that the software development process you adopt should

- provide only high-level guidance for team members, who should be able to direct their own work
- be simple and concisely described
- conform with the general principles of agile development
- focus on activities that legitimately add value
- be independent of particular toolsets, and
- be tailored according to needs for a particular project

Question

Match each phase of an AUP project to a description of what it involves.

Options:

A. Inception
B. Elaboration
C. Construction
D. Transition

Targets:

1. Conceptualizing an architecture for the required system
2. Validating an architecture for the required system
3. Building working software, with a focus on small releases
4. Testing and deploying the required system

Answer

In the Inception phase, the team establishes the initial scope of a project and conceptualizes a potential architecture for a system.

In the Elaboration phase, the system architecture conceptualized during the Inception phase is validated, and the team increases its understanding of system requirements.

In the Construction phase, the team builds working software on an incremental basis to meet the highest-priority needs of the customer.

In the Transition phase, the team completes system testing and deploys the system to a production environment.

Correct answer(s):

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

2. EssUP

The Essential Unified Process – or EssUP – defines a set of practices for software development. It derives these practices from UP, agile principles, and an approach to improving business processes known as Capability Maturity Model Integration – or CMMI.

In addition, EssUP emphasizes the concept of separation of concerns, or SOC. This refers to aspect-oriented thinking, in which specific concerns are identified and then addressed in order of priority.

Once you've identified and prioritized specific concerns, you can choose the EssUP practices that best address these concerns and incorporate them in your own development process.

In EssUP, a practice is defined as a proven way of approaching or addressing a problem. It has been tried and tested, can be communicated to others, and has reliable results.

EssUP defines five categories of technical and development practices. These are referred to as

Iterative Essentials

Iterative Essentials are practices focused on dividing the overall development process into incremental iterations, each of which is time-boxed and self-contained. This simplifies management and provides agility.

Architecture Essentials

Architecture Essentials are practices that center on creating robust, high-quality system architecture and on verifying this architecture through ongoing system-level testing, throughout the development process.

System architecture should also be refactored or redesigned as development proceeds, to adapt to changes and benefit performance.

Use-Case Essentials

Use-Case Essentials are practices that relate to requirements management, development, and system testing.

The customer and team members establish key system requirements and identify the value they're expected to add. These requirements are documented in the form of use cases, which are worded as sequences of actions. The specified requirements are also the basis for developing scenarios and test cases.

Component Essentials, and

EssUP focuses on component-based development. A team creates a model showing a system's components and how they'll work together. It then develops source code and unit tests for each component.

Model Essentials

Model Essentials are practices related to the use of models to represent a system's component structure and behavior, before coding and testing begin.

EssUP provides a lightweight set of modeling guidelines describing how models should be related and used.

In addition, EssUP defines three categories of supporting practices:

Product Essentials

Product Essentials relate to planning the evolution of software in the form of a series of product releases, each of which meets business needs.

EssUP stresses the need to include stakeholders in this process and to perform a stakeholder analysis, to ensure business needs are properly understood. It also emphasizes the importance of creating a sound business case for the product.

Process Essentials, and

Process Essentials are practices related to the processes that guide development.

These practices include continually improving and adapting processes, and creating a clear guide to describe the configuration and use of the selected process tools.

Team Essentials

Team Essentials are practices related to structuring a development team, and to how responsibilities are distributed among team members.

EssUP specifies that this information should be included in a team charter. It also provides guidelines on how team members should collaborate and organize their work.

Question

Which are key features of EssUP?

Options:

1. It outlines a development process in which iterations occur in each of four serial phases
2. It prescribes the use of a particular process
3. It provides a set of best practices for agile software development
4. It encourages the adoption of particular practices in accordance with an organization's chief areas of concern

Answer

Option 1: *Incorrect. AUP, rather than EssUP, outlines a development process in which iterations occur in each of four serial phases – including Inception, Elaboration, Construction, and Transition.*

Option 2: *Incorrect. EssUP provides a set of best practices, rather than prescribing the use of a particular process or development model.*

Option 3: *Correct. EssUP outlines a set of best practices that integrate UP, agile, and CMMI principles.*

Option 4: *Correct. EssUP emphasizes the need to identify and prioritize separate areas of concern, and to implement practices designed specifically to address these.*

Correct answer(s):

3. It provides a set of best practices for agile software development
4. It encourages the adoption of particular practices in accordance with an organization's chief areas of concern

3. Summary

AUP provides a process model for software development. Each project progresses through a series of four linear phases, but iterations occur within each phase.

EssUP defines a set of best practices for software development. It encourages organizations to select those practices that address their specific, prioritized areas of concern, and to incorporate these in their own development processes.

Crystal and DSDM

Learning Objectives

After completing this topic, you should be able to

- *recognize the features of Crystal*
- *identify the main principles of DSDM*

1. Crystal

Crystal is a family of methodologies, in which each methodology is named according to the differing colors of crystals of different hardness. It includes, for example, Clear, Yellow, Orange, Orange Web, Red, and Maroon.

The particular Crystal methodology you should apply depends on the complexity or "hardness" of a specific software development project.

As the size of a project team increases, a methodology with more prescribed steps and artifacts is required to keep control of the development process.

Note

Artifacts refer to documents required as outputs of specific steps and activities.

Similarly, the criticality of a project determines the extent to which a rigid process is required. Criticality refers to the potential of a system to do damage.

For example, if a system for air traffic control malfunctions, it could cost lives. A system for recording and editing music has much lower criticality – so processes for ensuring expectations are met don't have to be as rigid or fully defined.

Once a particular Crystal methodology is applied, it shouldn't be altered to include practices from the other Crystal methodologies.

For example, the Crystal Clear methodology – which is the simplest, and suitable for small projects – can't be expected to transition over time into a Crystal Orange project.

Instead, if a project's size and criticality increases, the appropriate methodology for its level should be adopted completely.

Typically, Crystal Clear is a suitable methodology for projects with small teams of up to six members.

Crystal Yellow is suitable for teams of seven to 20, Crystal Orange is for teams of 40 to 75, and Crystal Red is appropriate for teams of 75 or 80 members. Crystal Maroon is for very large projects, with 200 or more team members.

However, the criticality of a project should also influence the Crystal methodology you choose.

All the Crystal methodologies share a set of seven key principles. These emphasize the need for

frequent delivery

Crystal focuses on frequent delivery of smaller releases of working software, with deliverables produced at least every couple of months. On larger or more critical projects, the deliverables may be incremental. They may be designed for eventual integration into a larger release, but customers will still be able to provide feedback on them.

continual feedback

Crystal requires the members of a development team to meet frequently to update one another and provide feedback on project activities. The team is also expected to meet regularly with stakeholders, to ensure the project is staying on target to meet their expectations and to communicate progress.

constant communication

For small projects, it's expected that all team members will work in the same room to allow constant communication. For larger projects, team members should at least work in adjoining rooms.

In addition, all team members should have easy, ongoing access to the customer – or those responsible for defining system requirements.

safety

Like many other agile methodologies, Crystal emphasizes the need for all team members to feel they can express themselves safely and make and implement effective decisions.

In addition though, Crystal factors in the safety concerns of users, through the concept of criticality. If the end product may put users or their interests at risk, a project has higher criticality and a more rigid development process is required.

a focus on priorities

In any of the Crystal methodologies, team members are expected to know which areas of work are of the highest priority, and should be allowed to focus on completing these without interruption.

access to users, and

Crystal team members are expected to have access to the customer, or to one or more end users, throughout the development process. This ensures it's easy for the team to obtain feedback or clarify users' needs.

automated testing and integration

Crystal methodologies require the use of ongoing, automated testing at the component and system levels, the use of controls to support versioning, and frequent integration of system components.

Drill Down Home Page

Currently, two Crystal methodologies – Crystal Clear and Crystal Orange – are the most commonly used.

Page 1 of 2: Crystal Clear

In a Crystal Clear project, a team includes up to six members and these members are all expected to work in the same room.

The team includes only three specified roles. A sponsor handles project funding and is responsible for a project's approval, and a senior designer makes all final technical decisions. All other team members are programmers, and share responsibility for project management, business analysis, testing, and so on.

Page 2 of 2: Crystal Clear

A Crystal Clear team is expected to deliver a release of working software after each iteration of between 60 to 90 days.

Few artifacts – or documents – are required, although the team should define any artifacts it produces. The team is also responsible for defining its own set of coding standards and testing practices.

Page 1 of 3: Crystal Orange

The Crystal Orange methodology is appropriate for medium-sized projects, with a team of between 40 and 80 members.

It includes more emphasis on structuring the development process and on testing than Crystal Clear, and specifies a larger number of team roles.

For example, each subgroup of developers should include a tester. Roles such as Project Manager, Business Analyst, and Architect are assigned to specific team members.

Page 2 of 3: Crystal Orange

A Crystal Orange team is expected to release working software every 90 to 120 days, although it can choose to work in shorter iterations.

The team is responsible for defining its own coding standards, testing practices, and artifacts.

Page 3 of 3: Crystal Orange

Specific artifacts a Crystal Orange team is expected to deliver include

- requirements documentation
- a release sequence, or schedule
- a project schedule

- status reports
- a user interface – or UI – design document, if the project includes a UI
- an object model
- a user manual, and
- test cases

Question

Identify features of the Crystal methodologies.

Options:

1. The size and criticality of a project determines which methodology should be used
2. The number of features for inclusion in a software product determines which methodology should be used
3. Crystal Orange is suitable for small projects
4. Crystal Clear is suitable for teams of up to six members

Answer

Option 1: *Correct. Project size and criticality determine which of the Crystal methodologies should be used. As size and criticality go up, more rigid processes and a greater amount of written documentation are required.*

Option 2: *Incorrect. Project team size and the criticality of a project determine which of the Crystal methodologies should be used.*

Option 3: *Incorrect. Crystal Orange is suitable for medium-sized projects.*

Option 4: *Correct. Crystal Clear is suitable for small projects, with teams of six or fewer members.*

Correct answer(s):

1. The size and criticality of a project determines which methodology should be used
4. Crystal Clear is suitable for teams of up to six members

2. DSDM

The Dynamic Systems Development Method – or DSDM – is a framework for agile system development. Unlike other agile methodologies, DSDM reflects a business perspective rather than a technical one.

One of its main goals is ensuring the fitness of developed products for their intended business purposes.

Note

Because it lacks a more technical focus, DSDM isn't recommended for the development of safety-critical systems.

A DSDM project is roughly divided into three phases – activities that must occur before a project begins, those that make up the actual project lifecycle, and those that must occur once a project completes.

The Project Lifecycle phase is further divided into five stages – Feasibility Study, Business Study, Functional Model Iteration, Design and Build Iteration, and Implementation.

Within each phase, DSDM defines structured sets of activities and the relationships between these.

Graphic

A diagram represents the phases and activities of the Project Lifecycle phase in DSDM. The Feasibility Study and subsequent Business Study phases inform activities in each of the three other phases.

During the Functional Model Iteration phase, the steps include agreeing on a plan and then identifying, creating, and reviewing a functional prototype.

In the Design and Build Iteration phase, the steps are again agreeing on a plan and then identifying, creating, and reviewing a design prototype.

Steps in the Implementation phase are reviewing business needs, implementing the designed product, gaining user approval and creating guidelines, and user training.

DSDM requires a comparatively large number of artifacts and, for controlling these, the use of a configuration management system.

It also designates a large number of roles, with differing levels of authority, for team members.

However, it leaves room for flexibility in the way prescribed activities are incorporated in an overall development process.

A further important feature of DSDM is that it involves prioritizing system requirements. Those with low priority are then eliminated when necessary, to keep each phase of a project on schedule and within budget.

To prioritize requirements, a DSDM team categorizes them into four groups, represented by the MoSCoW acronym:

- requirements that MUST be met
- those that SHOULD be met if possible
- those that COULD be met provided doing this won't impact a project negatively, and
- requirements it WOULD be nice to include, but only if time permits

DSDM is based on nine key principles, which are especially relevant for agile software development. Five of these are

active user involvement

In keeping with its strong focus on the business purpose of a system, DSDM emphasizes the need for end users to participate in all stages of the development process. A business purpose can't be fully captured in an early product specification. Instead, input from end users should guide a team's decisions as the product evolves.

DSDM encourages the use of prototypes early in a project, to refine requirements and clarify stakeholders' expectations. It also encourages the use of workshops.

empowering teams to make decisions

DSDM relies on team members being given sufficient authority to make day-to-day decisions. This enables faster development times.

However, specified types of decisions – especially those with a potentially high impact on a project's success – must be escalated to those with greater authority.

frequent delivery of products

Progress in a project should be measured based on the delivery of work products. These products include documentation for team use, as well as software for delivery to customers.

use of fitness for business purpose as the chief acceptance criterion, and

What DSDM refers to as fitness for business purpose – in other words, the extent to which software meets the purpose for which it was developed – must be the main criterion for accepting deliverables.

iterative and incremental development

Like other agile methodologies, DSDM focuses on delivering small, incremental releases, each of which is validated by end users. This helps ensure a project stays on track toward meeting its business purpose.

Development must also be iterative, with repeated rechecking and revision of plans and work as a product evolves.

The four remaining DSDM principles are that

all changes during development are considered reversible

DSDM encourages change, with ongoing revision of plans and completed work as development proceeds.

requirements are baselined at a high level

In DSDM – unlike in many other agile methodologies – requirements are fixed at a high level. This is to provide a baseline against which progress can be measured. The baseline may change, but only if this is truly necessary and all stakeholders agree.

With input from end users, the development team is responsible for refining each high-level requirement and determining how best to implement it.

testing is integrated throughout the lifecycle, and

As in other agile methods, testing in DSDM is conducted throughout the development process. Each task should include a step for verifying or validating its output.

a collaborative and cooperative approach between all stakeholders is essential

DSDM relies on a high degree of collaboration among all stakeholders. All of the other DSDM principles depend on this.

Question

DSDM is based on nine principles.

Which are among these principles?

Options:

1. Delivering frequent releases to users
2. Integrating testing in all aspects of the development process
3. Fixing all product specifications at the outset of a project
4. Empowering team members to make decisions
5. Testing only at the system level
6. Basing acceptance of deliverables on fitness for business purpose
7. Viewing all changes made during the project lifecycle as reversible

Answer

Option 1: *Correct. DSDM focuses on the frequent, time-blocked delivery of small releases for validation by end users.*

Option 2: *Correct. A DSDM principle is that testing must be integrated as a step in every task, throughout the project lifecycle.*

Option 3: *Incorrect. The DSDM model involves baselining high-level requirements early in a project – but these are then refined and interpreted as a product evolves.*

Option 4: *Correct. DSDM relies on team members being given sufficient authority to make day-to-day decisions.*

Option 5: *Incorrect. DSDM focuses on integrating testing in all aspects of the development process.*

Option 6: *Correct. DSDM stresses that the main criterion for accepting a deliverable should be that it meets current business needs.*

Option 7: *Correct. DSDM requires constant rechecking and adjustment of plans and completed work, throughout the project lifecycle.*

Correct answer(s):

1. Delivering frequent releases to users
2. Integrating testing in all aspects of the development process
4. Empowering team members to make decisions
6. Basing acceptance of deliverables on fitness for business purpose
7. Viewing all changes made during the project lifecycle as reversible

Question

Which are remaining DSDM principles?

Options:

1. Equal authority of all team members
2. Active user involvement
3. Iterative and incremental delivery
4. A strict focus on meeting all planned requirements
5. Baselining of requirements at a high level
6. Ongoing collaboration among all stakeholders

Answer

Option 1: *Incorrect. DSDM assigns a variety of fixed roles, with differing levels of authority, to team members.*

Option 2: *Correct. DSDM stresses the need to involve end users in all stages of development.*

Option 3: *Correct. DSDM is geared towards iterative, incremental development, with continuous rechecking of work and frequent delivery of small releases for validation by end users.*

Option 4: *Incorrect. DSDM teams prioritize requirements, eliminating those with low priority when necessary to keep a project on schedule and within budget.*

Option 5: *Correct. In DSDM, high-level requirements are fixed near the outset of a project, to provide a stable baseline against which progress can be measured. These are then elaborated as a product evolves.*

Option 6: *Correct. All the DSDM principles rely on a high degree of collaboration among team members and with end users.*

Correct answer(s):

2. Active user involvement
3. Iterative and incremental delivery
5. Baselining of requirements at a high level
6. Ongoing collaboration among all stakeholders

3. Summary

Crystal is a family of methodologies for agile software development. The appropriate methodology depends on the size and criticality of a project. As team size and criticality go up, a more prescriptive and detailed development process is required.

DSDM is a business-oriented model for system development. It incorporates many agile principles, but focuses on ensuring the fitness of developed products for their intended business purposes. It also involves eliminating requirements with low priority when necessary, to keep a project on schedule and within budget.

Choose an Agile Approach

Learning Objective

After completing this topic, you should be able to

- *recognize factors to consider when moving to an agile software development methodology*

1. Factors to consider

Before implementing an agile software development methodology, you need to determine whether this type of methodology is suitable for your organization. This will depend on a number of factors.

Factors to consider include

- the structure of your organization
- project type
- the customer's role, and
- project management issues

2. Organizational structure

The structure of an organization is one of the factors that determines how easy it will be to transition into using an agile methodology.

For example, an organization's existing structure may or may not support the level of collaboration and the readiness to adapt to change that are key to agile development.

An organization's structure may be

hierarchical

In a hierarchical structure, roles and responsibilities are clearly defined. Authority figures are visible and take responsibility for delegating duties to employees of lower rank.

This type of structure isn't well-suited to an agile methodology because it restricts open communication and rapid innovation in response to change. Employees are largely expected to follow orders or set processes, and their work is finely managed.

cooperative, or

In a cooperative structure, employees tend to work in teams and collaboration is encouraged. There's less focus on levels of authority, and more on employees being flexible and working together to achieve goals.

An organization with this type of structure is likely to find implementing an agile methodology easier than one with a more hierarchical structure.

a balance of the two

Many organizations begin with a cooperative structure, but develop more hierarchical characteristics as they grow and become more established. When this occurs, a careful balance of the two structures may be established.

This type of hybrid structure is suitable for many agile methodologies, which also involve a careful mix of both cooperation and hierarchy. For example, agile development requires close collaboration among team members – but may also involve defining roles within a team based on a definite hierarchy.

Another factor to consider is how an organization typically approaches change.

An organization that focuses on controlling change to minimize the disruption it could cause to schedules and budgets may struggle with an agile methodology.

But if the focus is on welcoming change and routinely reacting to it when it occurs to improve customer satisfaction, an agile methodology will be easier to support.

Finally, an organization's employees must be able and willing to support an agile methodology.

Agile development depends on highly disciplined, skilled developers who can work well in teams. Simply putting an agile methodology in place won't be effective if existing employees lack the required competencies.

If you're planning a transition to an agile approach, you should also remember that employees will need time to adapt to the change, regardless of how they feel about their current work methods.

Graphic

A diagram shows the process and time that employees take to adapt to a transition. Transitions begin with a status quo, then denial, resistance, bargaining, compliance, testing, and finally acceptance. Energy levels fluctuate during a transition – at status quo and acceptance, energy levels are equal. During denial and compliance, energy levels are lowest and during resistance and bargaining, energy levels are at their highest.

To secure employees' support for a move to an agile methodology, you'll need to

- reinforce the need for the change
- train employees on how to implement the new methodology
- regularly update employees on whether and how the new methodology is working
- celebrate successes arising from the new methodology, and
- implement an effective reward system that recognizes employees' achievements

Question

Which factors should you consider in relation to an organization's structure before moving to an agile software development methodology?

Options:

1. How hierarchical the organization's current authority structure is
2. The organization's typical approach to managing change
3. The ability of existing employees to implement the methodology
4. The willingness to complete a project without customer involvement
5. The availability of highly trained managers to schedule, assign, and oversee all development tasks

Answer

Option 1: *Correct. An organization with a strictly hierarchical structure is likely to struggle to achieve the level of collaboration and adaptability required to support an agile development process.*

Option 2: *Correct. An organization that's used to reacting to change on an ongoing basis will find an agile methodology easier to support.*

Option 3: *Correct. Agile development depends on highly disciplined, skilled developers who can work well in teams. Also, employees must be given time to adjust to a new methodology and their support for it must be gained.*

Option 4: *Incorrect. Agile methodologies advocate customer involvement throughout a project's lifecycle.*

Option 5: *Incorrect. Agile methodologies focus on self-organizing teams and collaboration, rather than on detailed management of each team member's work.*

Correct answer(s):

1. How hierarchical the organization's current authority structure is
2. The organization's typical approach to managing change
3. The ability of existing employees to implement the methodology

3. Project type

The type of project a team needs to complete will help determine whether an agile methodology is suitable – and if so, which methodology should be used.

Before moving to an agile methodology, you should consider a project's

criticality and

A project's criticality refers to the severity of the impact its failure could have – on the organization running it and on the end users of the software produced.

It's generally best not to implement a new methodology for a project with high criticality, because changing existing approaches involves an element of risk.

However, it may be suitable to switch to an agile methodology for a highly critical project if, for example, traditional development methods have already been tried and have failed.

safety and security requirements

Safety and security requirements for software tend to be fixed. This contrasts with an agile development process, in which it's understood that requirements will change and be elaborated on during development. So if many safety and security requirements apply, it can be difficult to use an agile approach.

However, it's possible to incorporate some fixed project requirements in an otherwise agile methodology. With its focus on continuous testing and feedback, an agile development process may also help ensure that safety and security requirements are fully met.

As in the case of critical projects, you will need to weigh up the risks of adopting a new methodology against the possible benefits over traditional methods.

Although all agile methodologies involve developing software in an iterative and incremental manner, each methodology is different. The type of project and an organization's needs may help determine which methodology is suitable.

Some agile methodologies include

Extreme Programming – or XP

XP is an informal methodology that's especially suitable for small dynamic projects with constantly changing requirements. Its focus is on the practices of programmers, rather than on project management.

the Dynamic System Development Method – or DSDM

The emphasis in DSDM is on business requirements – delivering software within budget and on time. It's considered suitable only for types of projects with a specified list of characteristics, and includes complex processes and strict requirements for documentation.

Scrum

Like XP, Scrum is what's known as a lightweight method – with little focus on rigid processes or documentation. But like DSDM, its focus is on business value. Each requirement is prioritized based on its value to the customer.

Lean, and

Lean principles are suitable for incorporation with other agile methodologies. They focus on minimizing waste to increase efficiency and a project's return on investment.

Crystal

The Crystal family of methodologies facilitates the scaling of projects – with methodologies designed specifically for larger development teams. It also provides explicit support for the development of functions with high criticality.

Supplement

Selecting the link title opens the resource in a new browser window.

Job Aid

Access the job aid Comparing Agile Methodologies to review the strengths and weaknesses of different agile methodologies.

Some agile methodologies are easier to implement alongside existing development processes than others. For example, these include Scrum and Lean – which focus on process management rather than on specific technical practices.

XP, on the other hand, is programmer-centric and requires a greater degree of change for teams used to more traditional development methods.

You can choose to implement an agile methodology in different ways:

rigidly, with no deviations

Ideally, you should apply agile methodologies as prescribed, with no deviations. For example, Scrum prefers a rigid structure that's relaxed only once a project has been established.

However, before you apply a "by-the-book" methodology, you need to ensure it will support a particular project.

partially, tailoring it to specific needs, or

You can choose to implement only some aspects of an agile methodology or to customize the methodology based on project needs.

You may also choose to introduce elements of an agile methodology gradually, with a slow transition from a traditional development environment to one that's fully agile.

in combination with another methodology

You can use a "mix and match" approach to agile development, incorporating elements from different methodologies that you identify as suitable for supporting a particular project's goals.

This approach is useful because it's flexible and enables a gradual transition to a purely agile approach. However, projects may progress more slowly if teams have to adapt to multiple new methodologies.

Question

What should you consider in relation to a project's type before implementing an agile methodology?

Options:

1. The project's criticality level
2. The suitability of different agile methodologies
3. Whether all requirements can be fixed before development starts
4. The project's size, because agile methodologies are suitable only for small projects

Answer

Option 1: *Correct. It's best not to switch to a new methodology for a project with high criticality.*

Option 2: *Correct. Different methodologies are suitable for different project types. In addition, you may choose to combine methodologies or to tailor them based on needs for a particular project.*

Option 3: *Incorrect. If too many requirements – like those related to safety and security – are fixed at the outset, an agile methodology will be difficult to implement.*

Option 4: *Incorrect. Some agile methodologies – like Scrum and those in the Crystal family – can be scaled to larger projects.*

Correct answer(s):

1. The project's criticality level
2. The suitability of different agile methodologies

4. The customer's role

For an agile methodology to succeed, customers must be available and committed to participating in the software development process.

Note

A customer may be an internal one if software is being developed as part of a larger system. The customer may also be a contractor or an end user.

Traditionally, communication between a development team and the customer occurs through mediators, before development work starts and again after it completes.

Agile methodologies, however, depend on regular, direct interaction between programmers and the customer, who may form part of the development team.

A focus on customer involvement is common to all agile methodologies, although some stress it more than others. For example, XP and DSDM require continuous customer involvement throughout the life of a project.

Another consideration is that both the organization and the customer must be willing to accept flexibility, in requirements and through more open-ended contracts. This is because agility depends on accepting the potential for change as development work proceeds.

Question

What are some of the factors you should consider regarding the role of the customer when deciding whether to adopt an agile methodology?

Options:

1. Agile development requires a high degree of interaction between project teams and customers
2. Customers must be willing to accept flexibility in requirements and contracts
3. Communication with customers must occur through a third party
4. Customers should sign formal agreements that define their roles in projects

Answer

Option 1: *Correct. Agile methodologies require direct interaction between customers and project team members. So for this type of methodology to succeed, customers must be available and committed to participating in the development process.*

Option 2: *Correct. Agile methodologies depend on flexibility, with changes to requirements as development proceeds. So customers must be willing to accept this flexibility.*

Option 3: *Incorrect. Agile methodologies require regular, direct communication between a development team and the customer.*

Option 4: *Incorrect. Agile methodologies don't demand formal contracts and agreements about roles. They depend on both organizations and customers being willing to accept flexibility in the development process.*

Correct answer(s):

1. Agile development requires a high degree of interaction between project teams and customers
2. Customers must be willing to accept flexibility in requirements and contracts

5. Project management issues

A variety of project management issues affects the suitability of agile methodologies. These include factors associated with the composition of a project team, as well as with existing management processes.

Factors to consider in relation to the composition of a project team include

the size of the team

Because agile methodologies emphasize the need for face-to-face communication and interaction among team members, they're most suitable for small teams – typically of up to 15 developers.

However, some methodologies are scalable, allowing for the division of a much larger team into sub-teams.

team members' locations, and

Agile methodologies rely on close, ongoing interaction between team members. So they're difficult to implement properly if development team members are geographically dispersed.

whether multiple sub-teams will be required

If a scalable agile methodology is used and a development team will be divided into multiple sub-teams, it's vital to determine how to synchronize the efforts of the sub-teams.

If there isn't enough collaboration, agility – and project outcomes – will be compromised.

An organization's existing management processes may be either formal or informal, and this will help determine which agile methodology is suitable.

Becoming agile may also require changes to existing management processes and tools.

The types of existing management processes you should assess include those associated with

project management

In a formal project management process, a project is tightly controlled, with estimates, schedules, and milestones all used to track a project's progress.

In an informal process, there's less emphasis on using predefined criteria to control progress and fewer guidelines on *how* to achieve a project's goals.

requirements management, and

Requirements management may be rigid, with formal processes and highly detailed documents used to record and update all discussions related to a project's implementation.

Alternatively, it may be flexible and informal. For example, if initial requirements are flexible, a simple telephone call may be used to outline a project plan.

configuration management

Although most agile methodologies don't specify guidelines for configuration management, they do provide support for various of its elements. For example, agile development demands strict control over control code versions and over build and release versions. However, it's flexible regarding document and baseline control.

Different agile methods support varying degrees of formality. For example, methodologies such as DSDM are more formal and include more phases and documentation requirements than other methodologies, such as XP and Scrum.

Both DSDM and Crystal Orange have extended requirements phases known as elaborate requirements phases. DSDM and Feature Driven Development – or FDD – also include compulsory modeling phases, which XP and Scrum omit.

While testing is a major component of all agile development, it's particularly emphasized in DSDM – which requires a dedicated tester on each development team.

The Agile Unified Process – or AUP – is another highly formal method with rigorous documentation requirements.

Question

What are some of the project management issues to consider when deciding whether to implement an agile methodology?

Options:

1. The required size of the development team
2. The locations of team members
3. The required formality of management processes
4. The need for a high level of control over each team member's work
5. The need to manage comprehensive post-development testing

Answer

Option 1: *Correct. Agile methodologies are ideally suited to small development teams. If larger teams are required, it's more complex to implement an agile methodology effectively.*

Option 2: *Correct. Agile methodologies depend on close interaction between team members, so they're difficult to implement if team members are geographically dispersed.*

Option 3: *Correct. Different agile methodologies provide different levels of formality. For example, DSDM and AUP include more phases and documentation requirements than XP or Scrum.*

Option 4: *Incorrect. Agile methodologies emphasize the need for self-organizing teams, rather than for management of each team member's work at a fine level.*

Option 5: *Incorrect. In agile methodologies, comprehensive testing is integrated with development, rather than occurring once development completes.*

Correct answer(s):

1. The required size of the development team
2. The locations of team members
3. The required formality of management processes

6. Agile development tools

A variety of tools – including open-source tools that are free of cost – can be used to simplify the implementation of an agile methodology. However, it's important that you understand the methodology you've chosen properly before using these tools.

Some agile software development tools are

Eclipse

Eclipse is an integrated development environment – or IDE – that enables you to develop programs in various languages, such as Java and C#. Its functionality can be extended with a variety of plug-ins.

You can use Eclipse in an agile environment to assist in refactoring code. For example, if you change a class name, Eclipse will automatically convert all instances of the class – in both Java and non-Java files – to the new name.

Eclipse UML

Eclipse UML is a visual modeling tool that can be plugged into Eclipse. It's useful in an Agile environment because it's easy to use and has a lightweight code editor.

In addition, Eclipse UML enables you to generate models and to ensure all models and source code are automatically synchronized each time changes are made.

ANT

ANT is a Java build tool that you can use, for example, to create Java code, determine what program components are required, and create a Javadoc file for a compile system.

ANT is useful in an agile environment because it enables you to perform quick system rebuilds. You can also use ANT as an Eclipse plug-in so that you can build ANT programs and add ANT processes from within Eclipse.

Subversion, and

Subversion enables you to monitor changes to a program during development. It uses a client-server access method, so you can access a program you're working on from a computer in any location, provided it has Internet access.

Subversion is useful in an agile environment because it's capable of handling constant modifications to a system. In addition, it can be used to roll a system back to a previous state if necessary.

JUnit

JUnit is a Java test framework that can function as a standalone tool or as an Eclipse plug-in. It enables you to perform various, standardized unit tests – so even if different developers run the tests, they'll all follow the same procedures. In addition, JUnit enables you to duplicate test processes so that the same tests can be run under varying circumstances.

The Microsoft Solutions Framework – or MSF – is a vendor-based framework that's included in Microsoft Visual Team Studio 2010. It provides access to various concepts, models, and processes for use in a software development environment.

The MSF is useful especially for planning development in an agile environment. It includes features such as product backlog and iteration backlog workbooks, and reports.

Question

Which software development tool can you use to generate models in an agile environment?

Options:

1. Eclipse UML
2. Eclipse
3. JUnit
4. ANT
5. Subversion

Answer

Option 1: *Correct. Eclipse UML is a plug-in for Eclipse. As a version modeling tool, it's useful for developing models in an agile environment.*

Option 2: *Incorrect. Eclipse is an IDE that enables you to develop programs in various languages, including Java. It supports various plug-ins.*

Option 3: *Incorrect. JUnit is a Java test framework that enables you to perform unit tests. It can be used as a standalone tool or as an Eclipse plug-in.*

Option 4: *Incorrect. ANT enables you to build system Java code and to identify required program components. In an agile environment, ANT can be used to perform quick system rebuilds.*

Option 5: *Incorrect. Subversion enables you to monitor changes to a program during development. You can use it to make system modifications or to roll a system back to a previous state.*

Correct answer(s):

1. Eclipse UML

7. Summary

Before implementing an agile methodology, you need to assess various factors.

For example, you should consider whether an organization's structure, approach to change, and current staff will support an agile approach.

The type of project will help determine whether an agile methodology is suitable – and if so, which methodology should be used.

To support an agile approach, customers must be available and willing to participate in the development process.

Relevant project management issues to consider include the size of a development team, the location of team members, and the need to manage multiple sub-teams. Also, the required formality of management processes may determine which methodologies are suitable.

You can use various open-source tools to help implement an agile methodology. Examples are Eclipse, Ant, JUnit, and Subversion.

Moving to Agile Development

Learning Objective

After completing this topic, you should be able to

- *recognize the issues organizations face when applying agile principles in given scenarios*

1. Exercise overview

In this exercise, you're required to recognize the issues organizations face when switching to agile development methodologies.

This involves the following tasks:

- recognizing issues an organization may face in applying agile principles
- identifying issues that may affect the selection of an agile methodology, and
- identifying issues that may arise when implementing an agile methodology

2. Applying agile principles

An organization designs and develops software for the refrigeration industry. It's a highly cooperative organization that uses small teams, comprising engineering consultants and developers. Each team includes up to six members, none of whom are assigned specific team roles.

The teams currently use traditional software development methods, and have no experience in working with agile methodologies. Some of the developers work remotely.

Question

Which characteristics of the organization suggest it can readily support an agile development methodology?

Options:

1. Teams consist of skilled developers, who are given freedom to make their own decisions
2. The organization reacts to change as routine, with adaptations made whenever necessary
3. Delegation of duties from consultants to developers currently restricts teams from achieving goals efficiently
4. Employees must be able to adjust to a new methodology quickly

Answer

Option 1: *Correct. The success of an agile methodology depends on a skilled, self-organizing development team.*

Option 2: *Correct. Dynamic organizations that welcome change and adapt to it easily will find it easier to adopt agile methodologies than those that attempt to control and minimize change.*

Option 3: *Incorrect. Delegating duties is a sign of a hierarchical structure. This type of organization will have difficulty adapting to an agile methodology that focuses on the equal participation of all team members.*

Option 4: *Incorrect. Employees must be given time to adjust to an agile methodology, and their full support for its use must be gained.*

Correct answer(s):

1. Teams consist of skilled developers, who are given freedom to make their own decisions
2. The organization reacts to change as routine, with adaptations made whenever necessary

Question

Identify how the organization's project management style may affect the adoption of an agile methodology.

Options:

1. The organization's preference for small team sizes will enable it to adopt agile methods more easily
2. Because some team members work remotely, it may be difficult for them to meet regularly
3. The use of remote developers supports agile principles because these team members don't waste time traveling to work
4. Smaller teams mean members won't be able to cope with the demands of agile methodologies such as XP

Answer

Option 1: *Correct. One of the agile principles is effective communication among team members. This is best achieved in smaller teams of 15 or fewer people.*

Option 2: *Correct. An agile principle is that all team members must be able to communicate regularly and openly with one another. This is difficult if team members aren't co-located.*

Option 3: *Incorrect. One of the basic agile principles is the need for teams to be co-located so that face-to-face communication and interaction can occur among team members.*

Option 4: *Incorrect. Most agile methodologies, including XP, require the use of smaller teams so that all team members can communicate with one another effectively.*

Correct answer(s):

1. The organization's preference for small team sizes will enable it to adopt agile methods more easily
2. Because some team members work remotely, it may be difficult for them to meet regularly

3. Selecting an agile method

Question

Based on the organization's profile, match the agile methodologies it can implement with examples of their characteristics. Some methodologies may match more than one characteristic.

Options:

A. Extreme Programming – or XP
B. Lean
C. Kanban
D. Adaptive Software Development – or ASD

Targets:

1. Relies on self-directed teams
2. Provides general principles for streamlining development by eliminating wasted effort
3. Focuses on optimizing the flow of work through a development system

Answer

Both XP, ASD and Kanban all rely on self-directed teams in which the members make decisions based on their collaboration with one another, and with the customer.

Lean software development involves applying a set of general principles, rather than fixed processes or specific programming practices. Its focus is on streamlining the development process, improving its efficiency and eliminating wasted effort.

Kanban focuses on optimizing the flow of work through a development system. It does this by constantly limiting work in progress based on the real capacity of each subsequent point in the system.

Correct answer(s):

Target 1 = Option A, Option C, Option D

Target 2 = Option B

Target 3 = Option C

Question

Assuming the issue of developers working remotely can be overcome, which combination of methodologies could the organization choose for a software project that prioritizes technical requirements over business requirements.

Options:

1. Kanban
2. DSDM
3. Scrum
4. XP

Answer

Option 1: *Correct. Kanban could be incorporated with other agile methodologies such as XP and Scrum.*

Option 2: *Incorrect. DSDM focuses on business requirements over technical requirements, so it would not be a suitable choice. It also requires a large number of roles and artifacts, so would require a major change in work practice for this organization.*

Option 3: *Correct. Scrum could be used as a project management methodology in combination with Kanban, while using XP as a programmer-centric methodology.*

Option 4: *Correct. XP could be implemented as a programmer-centric methodology in combination with project management methodologies, Scrum and Kanban.*

Correct answer(s):

1. Kanban
3. Scrum
4. XP

4. Implementing an agile methodology

Phlogistix is an organization hired to develop software for a shipping company. The organization has a hierarchical structure, and encourages members to claim individual work ownership. A recent project was implemented using XP, but various problems arose.

Question

Identify actions that could ensure the successful implementation of XP for future projects.

Options:

1. Ensure close collaboration of team members with the customer
2. Focus on building a self-directed team of skilled developers
3. Encourage collective ownership of developed code by team members
4. Assign team leaders to delegate all development tasks
5. Use detailed documentation to record every aspect of the project's development

Answer

Option 1: *Correct. Most agile methodologies require a large degree of customer participation. In particular, XP demands continuous customer involvement throughout the life of a project.*

Option 2: *Correct. XP relies on self-organizing teams with the authority to make proactive decisions. It's likely to be difficult to implement in a hierarchical organizational structure.*

Option 3: *Correct. Collective code ownership is a key XP practice. Every team member must have full access to all the code for a release and can make any necessary changes to it.*

Option 4: *Incorrect. One of the features of XP is that team members function independently, with little direction from team leaders or higher-level managers.*

Option 5: *Incorrect. Unlike other agile methodologies such as Dynamic Systems Development Method – or DSDM, XP is an informal methodology that doesn't require a high level of project documentation.*

Correct answer(s):

1. Ensure close collaboration of team members with the customer
2. Focus on building a self-directed team of skilled developers
3. Encourage collective ownership of developed code by team members

Issues organizations face when applying, selecting, and implementing agile principles in given scenarios have been identified.