| [Print](javascript:window.print()) | [Close](javascript:parent.window.close()) |

**Course Transcript**

Managing Agile Software Development

**Implementing Agile Project Management**

| [1. Implementing Iterative Development](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#t3) |

| [2. Project Reports and Reviews](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#t6) |

| [3. Working with Documentation in an Agile Project](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#t8) |

| [4. Implementing Agile Management Tasks](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#t10) |

**Scaling Agile Projects**

| [1. Implementing Agile Project Scaling](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#t16) |

Implementing Iterative Development

Learning Objectives

*After completing this topic, you should be able to*

* *recognize how developers select tasks and perform iteration balancing*
* *recognize how to use meetings and charts to support communication and collaboration during iterative development*
* *recognize ways to manage change, quality, and risk during iterative development*

**1. Implementing iterative development**

Agile methodologies use the concept of *iterations*. An iteration is simply a length of time in which a small but complete portion of a project is completed, tested, and delivered to a customer.  
  
Structuring development work into a series of iterations ensures that a project provides regular output to a customer, with frequent small releases of working software rather than one final, large release.

Using iterations enables an agile project team to

* ensure customers are satisfied with each iteration outcome before beginning a new iteration
* mitigate risk, with previous iterations helping identify areas of uncertainty that need to be resolved
* receive feedback from customers and make changes to subsequent iterations as required based on this feedback
* more easily keep a project on course – because each iteration deals with a single but complete project feature, and
* if necessary, terminate a project smoothly at the end of an iteration

To implement an agile iterative development approach, the members of a team need to

* select the development tasks to complete in each iteration and, if necessary, balance the amounts of work in each iteration
* be colocated – that is, located in one place – and to engage in constant communication
* hold regular project meetings to keep things on track
* monitor a project's progress using tools such as charts and graphs, and
* manage risk, quality, and change throughout a project

It's necessary to determine what work to complete in each iteration – but the success of an iteration will depend on how well a team collaborates, monitors progress, and makes necessary adjustments to manage risk, quality, and change.

Question

Identify actions that can help ensure the success of an agile iterative approach to development.

**Options:**

1. Ensuring ongoing communication among team members
2. Performing project management tasks such as risk and quality management
3. Using distributed teams to encourage independent thinking
4. Evaluating progress only at the end of each iteration, to prevent loss of focus
5. Ensuring team members work in the same physical area

Answer

***Option 1:****Correct. The success of an agile iterative approach relies on project team members maintaining constant communication.*

***Option 2:****Correct. Performing project management tasks such as risk and quality management helps ensure the success of specific iterations and of a project as a whole.*

***Option 3:****Incorrect. To support an agile iterative approach, team members should be located in one place so that they can communicate effectively.*

***Option 4:****Incorrect. It's important to monitor progress – during regular team meetings and using tools like charts and graphs – throughout the development process, rather than only once an iteration completes.*

***Option 5:****Correct. Team members should be colocated so that they can collaborate easily.*

**Correct answer(s):**

1. Ensuring ongoing communication among team members  
2. Performing project management tasks such as risk and quality management  
5. Ensuring team members work in the same physical area

**2. Iteration tasks and balancing**

Identifying the tasks to include in an iteration is the first step in an iterative development approach. Typically the team determines how many tasks it can complete. The customer determines which tasks have the highest priorities and should be included.  
  
Team members then allocate tasks among themselves.

A developer on a team shouldn't take responsibility for more than two tasks at any one time. In this way, you help maintain the team's collective ownership of the work.

Iteration balancing involves redistributing tasks among different iterations to ensure that each iteration includes an appropriate amount of work.  
  
This may be necessary if the team initially underestimated the feature or user stories for an iteration – with the result that there's too much work for team members to complete within the allocated time. During iteration planning, the team may then need to reassign some user stories to a later iteration.

Iteration balancing may also be necessary if the team initially over-estimated features or user stories, and there's too little work for team members to complete. In this case, the team may take on tasks initially assigned to later iterations or request additional tasks from the customer. However, it should be careful not to risk failing to meet the iteration deadline.

Question

How should you perform iterative development?

**Options:**

1. Identify the tasks to include in an iteration
2. Redistribute tasks among different iterations
3. A developer on a team shouldn't take responsibility for more than three tasks at any one time
4. Have a sprint meeting

Answer

***Option 1:****Correct. During iteration development, the team determines how many tasks it can complete. The customer determines which tasks should be included and which have the highest priority.*

***Option 2:****Correct. Iteration balancing involves redistributing tasks among different iterations to ensure that each iteration includes an appropriate amount of work.*

***Option 3:****Incorrect. Developers shouldn't assume personal responsibility for more than two tasks at any one time.*

***Option 4:****Incorrect. You should have sprint meetings once you've reached the end of an iteration.*

**Correct answer(s):**

1. Identify the tasks to include in an iteration  
2. Redistribute tasks among different iterations

**3. Supporting iterative development**

A key agile principle is the need for ongoing communication and collaboration among team members. Typically, this requires that all team members work in the same room.  
  
This enables team members to communicate openly, without first having to locate one another or a suitable meeting area. It also promotes the idea of team ownership of a project.

In addition to working in the same physical environment, agile teams foster good communication through regular meetings and by using specific visual aids.

There are two common meeting types:

**stand-up meetings and**

Stand-up meetings are held mainly by teams implementing the Extreme Programming – or XP – methodology or the Dynamic Systems Development Method, known as DSDM. Participants are literally required to stand to ensure that these meetings don't last longer than they need to.  
  
Stand-up meetings are held frequently although they're not compulsory every day, and only programmers and team leads are required to attend. This is because if external parties such as customers attend, programmers may feel unable to discuss code freely without influencing external stakeholders' view of their progress.

**daily Scrum meetings**

Daily Scrum meetings are a feature of the Scrum methodology. They're similar to stand-up meetings, but are facilitated by a Scrum Master. The meetings are kept strictly to a duration of 15 minutes and are typically held in the same place at the same time every day. In contrast to stand-up meetings, Scrum meetings can be attended by all interested parties. However, only team members actually contribute during the meetings.

Both stand-up and daily Scrum meetings can be used to discuss design issues in a project. However, you can also choose to hold a separate design meeting at the start of an iteration. During this meeting, you can develop a high-level design that can be used to guide the project's development. You can then hold follow-up design meetings throughout the iteration.

An initial design meeting can also incorporate model storming. The models generated can be enhanced during later design sessions, before the team proceeds with code development.

Question

Which statements about the use of meetings by agile teams are correct?

**Options:**

1. Scrum teams hold daily Scrum meetings, led by the Scrum Master, to obtain feedback on a project's progress
2. XP teams hold very short meetings, attended only by programmers
3. Scrum teams hold bi-weekly progress meetings that are open only to development team members
4. DSDM teams hold regular, short meetings that all interested parties may attend

Answer

***Option 1:****Correct. Scrum teams hold daily Scrum meetings, which are led by a Scrum Master. The purpose of the meetings, which each last for a maximum of 15 minutes, is to obtain feedback on a project's progress – and on any problems that have been encountered – from all team members.*

***Option 2:****Correct. XP teams hold stand-up meetings, in which only programmers participate. The programmers stand during the meetings to help ensure they're kept as short as possible.*

***Option 3:****Incorrect. Scrum teams are required to hold daily Scrum meetings. All interested parties may attend these meetings, although only development team members contribute to the discussion.*

***Option 4:****Incorrect. DSDM teams, like XP teams, hold short stand-up meetings that can be attended only by programmers. This ensures that programmers can discuss code and their progress freely.*

**Correct answer(s):**

1. Scrum teams hold daily Scrum meetings, led by the Scrum Master, to obtain feedback on a project's progress  
2. XP teams hold very short meetings, attended only by programmers

Among the visual tools you can use to facilitate communication among developers are information radiators. These are visual representations of a project's progress.  
  
A visual radiator can be displayed in a common area to enable all team members continuously to check a project's progress and to update the status of tasks.

An example of an information radiator is a *taskboard*. This is a board where different aspects of a project including outstanding tasks, and progress made to date are displayed. Team members manually update the taskboard on an ongoing basis.

Graphic

*A taskboard is presented as a columnar chart with two rows or sections. Its four columns are headed Story, Tasks To do, In process, and Done. Space, which can be allocated to specific or concurrent project tasks, is provided beneath each heading – allowing team members space to annotate the chart where relevant or appropriate.*

Using a taskboard ensures a project's progress is visible to all parties and that, for example, tasks can't be forgotten or accidentally left incomplete.

You can use basic charts to make it easy for all team members to track progress. For example, XP generally advises the use of large, visible charts – such as posters put up in the areas where team meetings are held – to enable all programmers to track progress on a project.

You can also use more complex charts and graphs to monitor team and project progress. For example, agile teams use

**release burndown charts**

A release burndown chart or product backlog graph tracks the progress of each iteration or sprint against the amount of work remaining. In the chart, the vertical axis specifies the number of tasks remaining – or the number of story points still to be completed, and the horizontal axis records the number of iterations.  
  
For example, a release burndown chart may show that a project consists of four iterations and 80 tasks. It may reveal that at the end of the first iteration, the team has completed 20 tasks, and that it then experiences a slow period, marked by a spike or "burn" in the chart. Thereafter, the project proceeds smoothly. If tasks are added to an iteration, the chart may spike to display more outstanding tasks at the end of an iteration than at its beginning.

**iteration burndown charts, and**

Iteration burndown charts or sprint backlog graphs are used to track the actual hours of work done against the estimated hours/days within a specific iteration or sprint and can be used to review progress and plan a subsequent iteration strategy at the end of the iteration.

**velocity charts**

Burndown charts are also used to display the team's velocity across iterations.

Question

Match each type of chart or graph to the context in which it's used to support iterative development.

**Options:**

1. A chart in the form of a large poster
2. An iteration burndown chart
3. A product backlog graph

**Targets:**

1. An XP team uses this tool, which is placed in a common work area, to track progress
2. An agile team uses this tool to monitor work remaining in an iteration against progress
3. An agile team uses this tool to obtain an overview of a project's progress

Answer

*Large visible charts are typically used by XP teams to monitor project progress. They may take the form of posters attached to a wall where they can be viewed by all team members.*

*Typically, iteration burndown charts are used by agile teams to monitor progress in each iteration against the amount of work remaining.*

*A product backlog graph is a specific type of backlog graph used by agile teams to monitor progress across each iteration in a project.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

**4. Managing iterative development**

Agile methodologies prioritize sticking to iteration deadlines over completing the development of specific features.  
  
Tasks allocated for an iteration may take longer than anticipated to complete, or customer requirements may change once an iteration is already underway. In these cases, an agile team will either eliminate certain features or postpone work on them for a later iteration – rather than adjusting the deadline for the current iteration.

Customers may modify a plan either

**outside an iteration or**

Outside an iteration, a customer is allowed to modify requirements in any way, for example by adding or removing requirements from the plan. In this case, developers will re-estimate the schedule and work will continue. However, the customer will need to compromise in certain situations in order to meet the time constraints of the project.

**during an iteration**

It isn't generally advisable to make significant changes to requirements during an iteration. Certain methodologies, like Scrum, don't accommodate change at all at this point. In XP, this type of change may prompt a team to halt its current iteration and start planning for a completely new one. However, DSDM enables revisions during the objectives setting meetings for iterations, based on how features are prioritized.

In agile development, quality management occurs throughout a project to ensure the optimal performance of software. It may be conducted using various test methodologies.

Quality management is addressed differently in

**XP**

XP requires an in-depth testing process during which a product must undergo multiple unit tests and obtain a 100% pass. Automatic unit tests can be performed using XUnit, and testing is prioritized even over code development. Integration testing using build and smoke tests is performed on a daily basis. In addition, various functional tests must be performed and a peer review of code is required.  
  
XP teams also conduct User Acceptance Testing, or UAT, during the development stage. For example, customers monitor all project stages, with various tests performed throughout the project. A formal review is conducted at the end of the project.

**DSDM, and**

DSDM specifies various quality criteria for all products that are developed and advises the use of peer reviews during testing. In addition, DSDM teams conduct User Acceptance Testing, or UAT. These tests are conducted throughout a project, with formal reviews during each iteration and at the end of the project.

**Scrum**

Scrum demands daily testing in the form of build and smoke tests. In addition, Scrum teams implement User Acceptance Testing, or UAT, throughout a project, with a formal review once a project is completed.

Generally, agile methodologies reduce risk because of their iterative approach. However, risk exists in any project – and it's important to manage this risk to optimize a project's chance of succeeding.

Some agile methodologies, like DSDM, provide specific guidelines for managing risk, while others are less prescriptive. For any agile team though, key risk management activities include identifying and analyzing risks, implementing plans to mitigate specific risks as necessary, and monitoring projects continuously for potential problems.

Agile methodologies identify five major risks in all practices. These risks are

* feature creep – additional, unnecessary, or overly complex functionality
* short-changed quality – compromised function or feature design
* gold plating – work exceeding a project feature or function's value or need
* unrealistic estimates, and
* friction between customers and developers

Projects may also be at risk due to inefficient project designs. Practices like modeling can help prevent this – although modeling isn't specifically required by all agile methodologies.

To mitigate risk, an agile team can use

**shallow iterations**

Shallow iterations, or analyses of entire projects wherein only broader, more superficial details are considered, require less commitment to the functionality of specific project functionalities. In other words, they are not greatly affected by negative risk.  
  
Some methodologies require both deep and shallow iterations. For example, Crystal Orange recommends the use of three deep iterations per development cycle. These provide either a mock-up or a fully functioning system that enables customers to provide feedback.

**prototyping, and**

A software prototype is a working example of a program.  
  
Prototyping is recommended by DSDM. It can be used to demonstrate business processes, user functionality, and system performance and capacity. You can also develop prototypes to illustrate design concepts, and to give customers an idea of how an end product will function.

**spike solutions**

Spike solutions are basic programs that can be used to identify and resolve potential problems in a program. These programs should be built to focus only on a specific problem. They're useful because they enable you to increase the reliability of user story estimates, as well as to mitigate technical risks.

Question

Which statements about how different methodologies manage change, risk, and quality are correct?

**Options:**

1. XP may handle change during an iteration by creating a new iterative plan, and performs various unit tests throughout the project to ensure maximum quality
2. Scrum doesn't accommodate change during iterations and monitors quality by performing daily smoke tests
3. DSDM monitors product quality by performing daily build tests and managing risk using shallow iterations
4. XP doesn't accommodate change during an iteration and specifies prototyping as a method of managing risk

Answer

***Option 1:****Correct. XP accommodates change during iterations but may require that the iteration begin again to implement the change. XP teams monitor quality by performing unit tests throughout a project.*

***Option 2:****Correct. Scrum doesn't accommodate modifications to a plan during an iteration. In addition, Scrum teams perform quality testing by performing daily smoke and build tests, as well as various UAT procedures.*

***Option 3:****Incorrect. DSDM maintains product quality through peer reviews and UAT. Also, it uses prototyping rather than shallow iterations.*

***Option 4:****Incorrect. XP accommodates change during iterations but may require that the iteration begin again to implement the change. Prototyping is recommended by DSDM as a method for managing risk.*

**Correct answer(s):**

1. XP may handle change during an iteration by creating a new iterative plan, and performs various unit tests throughout the project to ensure maximum quality  
2. Scrum doesn't accommodate change during iterations and monitors quality by performing daily smoke tests

**5. Summary**

Agile methodologies use an iterative approach to ensure frequent, small releases of software to a customer. For iterative development to succeed, team members need to divide work among discrete iterations, collaborate closely, and manage project quality, risk, and change.  
  
During iteration planning, a team and customer need to determine which tasks to complete during an iteration. After initial planning, the team may also need to balance iterations by distributing tasks more evenly among them.  
  
To facilitate communication, agile team members should work in the same physical area and hold regular meetings. They may also use visual aids like charts and graphs to monitor a project's progress.  
  
Agile teams can monitor quality by performing build and smoke tests and holding peer reviews. Examples of methods for managing risk are prototyping, spike solutions, and shallow iterations. Different methodologies use different approaches to managing changes that occur, during or between iterations.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#top)

Project Reports and Reviews

Learning Objectives

*After completing this topic, you should be able to*

* *recognize different charts that provide feedback during a project*
* *recognize the type of feedback provided by reviews in agile projects*

**1. Obtaining feedback from metrics**

It's important to monitor and report progress when working on an agile project. You can do this using a combination of methods.

Typically, agile teams rely on two main types of feedback. These include feedback derived from charts, reports, and metrics, and feedback that comes from people, in the form of reviews.

Binary status reporting is a method you can use to establish what has been done and what still needs to be done – or which features are complete and which are still in progress – in a project.

This method is more accurate than methods that estimate the percentage of work that has been completed because percentage values are difficult to approximate.

There are two agile methodologies in which binary status reporting is important:

**Scrum and**

Scrum uses sprints to manage and evaluate the progress of work. Typically programmers are evaluated based on what they have delivered at the end of each sprint, at which stage features will be noted as either complete or incomplete.

**FDD**

Feature Driven Development, or FDD, identifies series of milestones in the work required to develop each product feature. Progress in developing each feature can then be tracked by determining which milestones have been passed and which must still be reached.

When tracking the progress of a project, timeframes are important. Velocity is a core concept that you should use to manage agile projects. It represents the amount of work that a team can complete within a certain amount of time. Velocity is represented as a number of story points or ideal days that can be completed during each iteration.

Note

*The total number of story points or ideal days assigned to a project is a measure of the estimated total amount of work involved.*

After each iteration is complete, it's important that you recalculate velocity and use the result to adjust your plan for the next iteration, and for the project as a whole.

This is important for maintaining the integrity of project planning. Once this has been done, charts such as burndown and burnup charts can be created to provide information on outstanding work.

You can use a release burndown chart to measure progress by determining the amount of work that remains at the beginning of each iteration in a project. This is useful for visually representing the progress that is being made on a project.

Typically, a release burndown chart contains iterations on the horizontal axis and story points or ideal days on the vertical axis. The graph indicates how much work remains at any given stage in the project, as well as the rate of the progress that's being made.

By continually charting progress, you can determine whether allocated work is likely to be complete within the timebox that has been assigned. If this seems unlikely, you can then make the necessary adjustments.

If work is added to a project, or an inaccurate estimate of the effort involved in the project was made initially, your chart might depict a burnup. This is indicated as an iteration that ends with more work to do than when the iteration began. This is a good indication that adjustments need to be made to the project plan.

You can choose to draw release burndown charts as bar charts. This type of chart displays velocity and scope separately, with each bar indicating the amount of work that remains at the start of each iteration.

If work is added to the project, the bottom of the bar will dip below the horizontal axis.

When drawing or interpreting release burndown bar charts, you need to remember that

* when work is completed, the top of a bar will be lowered
* re-evaluation of work will move the top of a bar up or down
* if work is added to the project, the bottom of the bar is lowered, and
* if work is removed from the project, the bottom of the bar is raised

Examples of other tools you can use to track the progress of an agile project are

**parking-lot charts**

Parking-lot charts consist of boxes that represent each theme within a project. Each box should depict how many stories the theme contains, the number of ideal days assigned to develop these stories, and the percentage of the stories that have already been completed.  
*A parking-lot chart comprises three boxes labeled project themes; Testing, Reporting, and Security. Testing comprises 8 stories and 36 story points, Reporting 12 stories and 41 story points, and Security 4 stories and 18 story points. The bottom third of each box consists of a progress bar indicating completion levels. Swimmer Demographics reflects 50 percent, Reporting 100 percent, and Security 33 percent.*

**burnup charts, and**

Burnup charts are similar to release burndown charts. However, they indicate both the amount of work completed and the amount of work remaining in each iteration. They also more clearly depict any changes that have been made to the total amount of work throughout the project.  
*Reflecting iteration work completed and work remaining, four vertical bars along a horizontal time axis measure their time in Story points against a vertical axis allowing positive and negative values. Incorporating these values, each bar is shaded to the percentage degree to which it is complete.*

**feature set milestones**

If you use FDD, you typically categorize work into feature sets. You should then use well-defined milestones to measure the progress that's being made within each feature set. Milestones are tangible and so can be recorded as either complete or incomplete. So you can determine how much progress has been made in developing a feature set based on the most recent milestone reached. Six milestones are defined for each feature which include three within Design by Feature activity and three within the Build by Feature activity.  
*Milestone 1 and Milestone 2 are charted, marked, and divided into time units on a linear horizontal axis. Milestone one, containing three features or one feature set, is complete and Milestone 2 reflects two features of a further three-feature set completed.*

During an agile project, you'll also need to track the progress of areas that don't relate directly to the completion of work. There are three main metrics that you can use to do this:

**acceptance tests passed**

You should track the number of acceptance tests that you run and that are passed at the end of each iteration. Depending on how many tests a feature has passed, you can determine whether it's ready to be released.

**unit tests run, and**

You should monitor the number of unit tests run if your team has just started using automated unit testing or if you need to encourage or emphasize the use of testing. The unit tests run metric tracks the number of automated unit tests that have been run during the last build of the day.

**estimated versus actual work done**

It's important to monitor how a team's estimates compare to actual productivity. By tracking estimated performance against actual work that's completed, you can improve the accuracy of subsequent estimates.

In addition to standard metrics, different agile methodologies use specific techniques to report on project status. For example, Crystal Orange specifies status reports as specific deliverables within a release, and Scrum emphasizes the role of daily Scrum meetings in reporting and tracking progress.

Question

An agile team can use a variety of chart types and metrics to track progress.  
  
Match each metric with the feedback it provides.

**Options:**

1. Release burndown charts
2. Parking-lot charts
3. Burnup charts
4. Charts tracking milestones
5. Acceptance tests passed

**Targets:**

1. The amount of work that remains at the beginning of each iteration in a project
2. The percentage of a theme that has been completed
3. The amount of work completed and remaining in each iteration
4. The progress made in developing a feature set
5. The release readiness of a feature

Answer

*You can use a release burndown chart to measure progress by determining the amount of work that remains at the beginning of each iteration in a project. This is useful for visually representing the progress that is being made on the project.*

*Parking-lot charts depict boxes that represent various themes within a project. Each box indicates the number of stories the theme contains, how many ideal days are allocated to the stories, and the percentage of the stories that are complete.*

*Burnup charts illustrate the amount of work completed and the amount of work remaining in each iteration. They also indicate any changes that have been made to the amount of work in the project.*

*FDD uses defined milestones to measure the progress made in developing each feature set in a project.*

*By tracking the number of acceptance tests that you run and that pass at the end of each iteration, you can determine whether a feature can be released.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

Target 5 = Option E

**2. Obtaining feedback from reviews**

You can use reviews to obtain feedback from various people about agile projects. You should do this by frequently requesting and providing opportunities for feedback from the members of your team. It's also important to obtain feedback from the customer and other stakeholders.

Agile methodologies encourage the use of open workspaces – which make it easier for team members to communicate – and regular meetings.

Review sessions are also important because they provide opportunities for the team to evaluate what has been working, as well as what hasn't.

During meetings, you should discuss

* how the customer feels about the present state of the system
* how all programmers feel about the present state of the system
* the general status of the team
* which practices and processes are being used, and
* the status of the project as a whole

Different methodologies generally conduct reviews in different ways and use different terms to describe them.

The Scrum methodology uses three main types of reviews:

**sprint review**

Sprint reviews are four-hour meetings in which the team presents the completed features to the product owner and stakeholders. The features are run on a team workstation and executed from a nearby server. A team member outlines the sprint goal, the product backlog commitment, and what was achieved. The team then discusses how the sprint went.  
  
From the meeting, stakeholders provide comments on the functionality and any changes that they require in that functionality or in outstanding work in the product backlog.

**sprint retrospective, and**

During a sprint retrospective, the entire project team meets to review progress. The team's customer might also join the meeting. Generally, this type of meeting lasts up to three hours. It's used to obtain feedback from the team and to identify activities that should be performed in the next sprint.  
  
This type of meeting is opened by the Scrum Master, who asks each team member to describe what worked well in the previous iteration, and what needs to be improved in the next iteration. The issues raised are then discussed, and solutions are identified.

**reflection workshop**

In a reflection workshop, the project team, customer, and any other people who might have input on the project identify effective practices, as well as problems that arose and possible solutions for them. Often a flip-chart with a table with three sections – Keep These, Problems, and Try These – is used to focus participants and record the ideas generated.  
  
In XP, a review of this type – lasting between one and two hours – is held at the end of each iteration. In the Crystal Orange methodology, a reflection workshop is held after each increment or release.

Question

You can use reviews to obtain feedback about a project from team members and other stakeholders.  
  
Match each type of Scrum review with the feedback it provides.

**Options:**

1. Sprint review
2. Sprint retrospective
3. Reflection workshop

**Targets:**

1. Comments about functionalities and the changes that are required
2. Ideas about what worked in the previous iteration, and improvements needed for the next iteration
3. Thoughts on what went well in a project, and what problems were encountered

Answer

*In sprint reviews, the team present completed features, the sprint goal, the product backlog commitment, what was achieved, and how the sprint went. Stakeholders provide comments about functionalities and any changes that they require.*

*Sprint retrospectives are used to obtain feedback from the team about what worked well in the previous iteration, and what needs to be improved in the next iteration. The issues that are raised are then discussed, and solutions are identified.*

*Most reflection workshops use a diagram that stimulates discussion about what worked in a project, what problems arose, and solutions to any problems.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

**3. Summary**

You should monitor progress when working on agile projects. You can do this using metrics and reviews. Release burndown charts measure progress by determining the amount of work that remains at the beginning of each iteration. You can also use parking-lot charts, which show the percentage of a theme that has been completed, and burnup charts, which indicate the amount of work completed and remaining in each iteration. If you use FDD, you can track progress by measuring milestones. Acceptance tests passed, unit tests run, and estimated versus actual work done are additional metrics that can be useful.  
  
You should frequently request and provide opportunities for feedback from the members of your team. During meetings, you should discuss how the customer and programmers feel about the present state of the system, the status of the team, the project, and which practices and processes are being used. The Scrum methodology uses three main types of reviews – the sprint review, sprint retrospective, and reflection workshop.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#top)

Working with Documentation in an Agile Project

Learning Objectives

*After completing this topic, you should be able to*

* *match typical agile project documentation with descriptions*
* *identify best practices for documentation in an agile project*

**1. Agile documentation**

The deliverables in an agile project can include documents, source code, and models.  
  
In this context, a document is used to exchange and record information – unlike actual source code, which contains the series of instructions used to develop a software system.  
  
Models are abstract examples of project problems or solutions. A team may integrate models into a document, or even use a model to develop source code.

Both traditional and agile methodologies recognize the need to document aspects of a project, such as requirements and estimates. Traditional methodologies interpret documentation as a blueprint for a project. Agile methodologies, however, view the time it takes to create documents as time a team could have spent developing code – so they emphasize keeping documentation activities to a minimum.

In a traditional environment, project documentation follows a chain of command, with each change recorded and approved by a higher authority before it's implemented.  
  
Although this provides stability, it can be time-consuming and isn't feasible in an agile environment, where teams are encouraged to accept change and rapidly to implement the solutions best-suited to a project.

Each of the agile methodologies differs in the amount and type of documentation it requires. For example, Extreme Programming – or XP – teams create very little documentation, whereas Dynamic Systems Development Method – or DSDM – demands a high level of documentation.  
  
In all instances, however, the level of documentation required is far lower than that demanded by traditional methodologies.

Some key points to consider before generating an agile document are whether the document will be

* quick and easy to maintain
* used regularly by team members, and
* necessary to the success of the project

Ideally, all documentation for an agile project should

**provide maximum value**

A document is considered agile if it provides a good return on investment, or ROI. In other words, the benefit of the document to a project should exceed the time and effort spent in developing and maintaining it.

**be financed by stakeholders**

In an agile document, the total cost of ownership is declared to project stakeholders, who in turn agree to bear the cost of developing and maintaining the document.

**focus on vital aspects and be concise, and**

Agile documents shouldn't be lengthy and should record only the most important aspects of a project. This ensures that valuable time isn't wasted. In addition, it's accepted that the content of an agile document doesn't have to be perfect and may contain flaws.

**be specific for every project customer**

Every document developed in an agile project should be tailored to the specific customer's needs. This ensures that documents focus on a key goal. For example, a project targeted at maintenance developers will require only system documentation.

Question

Match each type of methodology with descriptions of the roles played by documentation in each. Each type of methodology may match to more than one description.

**Options:**

1. Agile methodology
2. Traditional methodology

**Targets:**

1. Used to record only the essential aspects of a project
2. Used to obtain customer approval before changes are implemented
3. Used to guide all project tasks
4. Ideally used only when the benefit of a document will outweigh the effort required to create and maintain it

Answer

*Agile documents record only the most important aspects of a project. This saves time, which team members spend developing code.*

*Traditional methods require that all changes to a project be documented, with changes approved by customers and other stakeholders before they're implemented.*

*Documentation in traditional projects serves as a guideline for project activities. Agile approaches uses documentation simply to maintain official records of the most important aspects of a project.*

*Agile teams view documentation activities as necessary but as negatively impacting productivity. So an emphasis is on creating only documents that are really required or beneficial.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option B

Target 4 = Option A

**2. Types of agile documentation**

You need to develop certain project documentation regardless of the type of agile methodology you implement. There are various types of agile documents, for example documents that are used during the course of a project and documents developed when signing off a project. Each document type has a particular target audience and serves a specific purpose.

In contrast to traditional development environments, agile teams take a lightweight approach to the documents they produce, which include

**mission/requirements documents**

Agile methodologies recommend an incremental approach to documenting requirements, with XP recommending just a short mission statement known as a metaphor as an initial statement of requirements, with index cards used to record features or user stories during planning. FDD on the other hand requires a model of the system at project initiation while ASD recommends producing 3 specific documents – the project vision, the project data (or overview) sheet and the product specification outline.

**design documents**

Agile methodologies vary in their specification for design documents, although here too the principle of “just enough” applies. Agile XP teams apply a very informal approach to design with model or design storming within iteration planning and development, and they rely on tests and well written code to fulfil design document requirements. On the other hand, FDD requires sequence diagrams to expand on the system model, and DSDM calls for specific design documents.

**rationale and structure documents, and**

Agile teams often maintain a short document outlining the rationale of the system being developed and its overall structure. This is useful for new team members, external interested parties, and for system maintenance teams. This document can also serve as a design document for the project.

**user documentation**

Writers of end-user documentation should apply the agile principle of "just enough". In an agile environment, user stories can be leveraged to produce task-oriented and focused user documentation in an incremental manner at the end of each release cycle.

Providing other teams, customers, or stakeholders with a formal document – typically with the aim of sharing information – is referred to as a document *handoff.*  
  
Generally, few document handoffs occur in an agile process. This is because agile developers view face-to-face meetings as a more efficient method of communicating than sharing documents.

However, document handoffs can't be avoided completely. For example, you may need to provide updates to team members located off-site, or provide status updates to stakeholders who aren't readily available for meetings. In these cases, you should try to make handoffs as effective as possible.

In an agile environment, it's considered best to

* minimize document handoffs, instead encouraging project members to communicate in person – for example, using video-conferencing if face-to-face meetings aren't possible
* use additional means of communication during a handoff to help prevent misunderstandings, and
* ensuring the documents transferred to others meet agile requirements – covering only what's necessary in a concise way

Question

Match each agile document type to its description. Some options will not be used.

**Options:**

1. Mission and requirement documents
2. Design documents
3. Rationale and structure documents
4. User documentation
5. Operations documentation

**Targets:**

1. Documents that may comprise just a short mission or vision statement, or include a project data sheet
2. Documents that may range from simple informal models, to specified sequence diagrams, or to specific design documents
3. Short documents that outline the reasons why a system is to be developed and its overall structure
4. Focused, task-oriented documentation produced for end-users in an incremental manner at the end of each release cycle

Answer

*Agile methodologies recommend an incremental approach to documenting requirements, with XP recommending a short mission statement with index cards used to record features or user stories during planning. FDD requires a model of the system at project initiation while ASD recommends producing the project vision, the project data (or overview) sheet and the product specification outline.*

*Agile methodologies vary in their specification for design documents. XP calls for a very informal approach to design with model or design storming within iteration planning and development. FDD requires sequence diagrams to expand on the system model, and DSDM requires a number of specific design documents.*

*Agile teams often maintain a short document outlining the rationale of the system being developed and its overall structure. These documents can also serve as a design document for the project.*

*The agile principle of "just enough" is applied to end-user documentation. In an agile environment, user stories can be leveraged to produce task-oriented and focused user documentation in an incremental manner at the end of each release cycle.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

**3. Agile documentation best practices**

In an agile environment, various principles can help you create required documentation efficiently and so that it suits the needs of agile teams.

General best practices for creating and managing agile documentation include

**documenting actions rather than ideas**

A good agile practice is to defer creation of documentation as much as possible until the end of a project so that you document only what has already been done or created. If you create documentation too early in the process, you may find that you need to update documents too frequently. However, you can keep notes throughout a project so that you don't forget important information. You can then use these notes to compile a comprehensive document at the end of the project.

**always confirming that a document is really needed before creating it**

Before developing a document, you should confirm that the people who've requested it can explain why they want it, and how they intend using it. They must also be willing to finance the document's creation and the resources needed to maintain it.

**ensuring there's a simple, efficient way to maintain system documentation, and**

You can save time and money by maintaining system documentation using software-based modeling tools. These tools enable you to reverse-engineer code and to modify it using various views.

**positioning content effectively within documents**

A good agile document contains information that has been placed effectively. For example, when developing a document, you should determine whether a design model is best positioned with the corresponding code or in a document on its own.

Additional best practices include

**prioritizing and estimating tasks associated with creating documentation**

An agile team is encouraged to approach the development of a required document in the same way as other project requirements. For example, the team should prioritize creating the document along with other tasks, and estimate the time and effort required to develop the document.

**developing documents in stages**

Documentation should occur in stages so that you can obtain feedback on parts of a document before completing it. This ensures that you create a document suited to your audience, and contains the information that this audience requires.

**focusing on customer requirements**

A good agile document focuses on customer requirements and on meeting customer needs. This is in keeping with the overall focus of an agile development approach.

**using up-to-date models, and**

Before starting to create documentation, you should consider which models are available. Models that are up to date are usually valuable and can be used as the basis of your documentation.

**using documentation to record information rather than to communicate**

In agile methodologies, most communication occurs face-to-face so that ideas can be discussed. So the purpose of documentation isn't to enable communication. Instead it's to record information about a project – for example, what actions were taken and the results this had.

You should also ensure that documents are

* created only when they have a specific purpose and will help fulfill a project goal
* simplified – containing just enough information to provide context for the reader
* publicly available so that they are accessible and can be used frequently
* written by experienced writers who are able to write and arrange technical content effectively
* composed of unique information not repeated in other documents, and
* updated only when absolutely necessary, to save time

Question

What are some best practices to follow when creating documentation for an agile project?

**Options:**

1. Develop documents in stages
2. Manage the work required to create documentation in the same way as other project work
3. Update documents only when necessary
4. Develop documentation at the start of a project
5. Document all project tasks, including as much detail as possible

Answer

***Option 1:****Correct. You should develop a document in stages so you can obtain feedback on parts of the document before completing it. This enables you to create a document that's relevant to your audience.*

***Option 2:****Correct. An agile team should prioritize the work required to develop a document in the same way as it prioritizes other project tasks. It should also estimate the time and effort required to develop the document.*

***Option 3:****Correct. An agile team should update documents only when absolutely necessary. This saves time that can be spent on more important tasks, such as code development and testing.*

***Option 4:****Incorrect. An agile team should develop documentation closer to the end of a project. This ensures that it documents only what has actually been done, which saves time.*

***Option 5:****Incorrect. An agile team should create a document only when it will fulfill a specific goal, or is directly related to the success of a project. Also, each document should include as little detail as possible. It should focus on covering only what's really necessary.*

**Correct answer(s):**

1. Develop documents in stages  
2. Manage the work required to create documentation in the same way as other project work  
3. Update documents only when necessary

**4. Summary**

Agile development methodologies require less documentation than traditional methodologies. They focus on creating project documents only if these will provide value and are necessary to support a project's goals.  
  
Among the best practices for creating agile documents are justifying the need for all documents before creating them, and creating documents in stages.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#top)

Implementing Agile Management Tasks

Learning Objectives

*After completing this topic, you should be able to*

* *implementing agile management tasks*
* *monitor and communicate team progress*
* *work with documentation in an agile project*

**1. Exercise overview**

In this exercise, you're required to recognize how agile methodologies implement various management tasks.

This involves the following tasks:

* identifying methods for supporting an iterative development approach
* recognizing methods for monitoring and communicating team progress, and
* identifying principles for agile documentation

**2. Task 1: Managing agile projects**

Various practices and activities are needed to support an Iterative development approach.

Question

Identify how task and iteration balancing are performed as part of iterative development.

**Options:**

1. Team members should select a maximum of two tasks at any one time
2. Extra tasks should be removed if user stories are overestimated in an iteration
3. Team members take on additional tasks if user stories are overestimated
4. Different team members are held fully accountable for the completion of different tasks

Answer

***Option 1:****Correct. Although developers may select tasks that they're personally responsible for, they need to ensure they don't undermine the team's collective ownership of the work. To do this, they should restrict their tasks to a maximum of two.*

***Option 2:****Incorrect. If team members have completed all their tasks before the iteration deadline, they can take on additional tasks initially assigned to a later iteration, or request additional tasks from the customer. Extra tasks should only be removed if user stories have been underestimated and there is too much work for team members to complete within the allocated time.*

***Option 3:****Correct. If team members overestimate the time it takes to complete all the tasks from the current iteration, they can take on additional tasks initially assigned to a later iteration, or request additional tasks from the customer, as long as this will not affect the iteration deadline.*

***Option 4:****Incorrect. Although each team member is responsible for particular tasks, it's the entire team's responsibility to ensure that tasks are completed.*

**Correct answer(s):**

1. Team members should select a maximum of two tasks at any one time  
3. Team members take on additional tasks if user stories are overestimated

Question

How do different agile teams use meetings to support iterative development?

**Options:**

1. Extreme Programming – or XP – teams hold short meetings that are attended only by programmers
2. A Scrum Master hosts a daily meeting that's attended by all team members and interested parties
3. Scrum teams hold weekly meetings that are attended by programmers
4. Dynamic Systems Development Method – or DSDM – teams hold daily stand-up meetings that are attended by all interested parties

Answer

***Option 1:****Correct. XP teams hold stand-up meetings during which members stand in order to keep meetings short. Stand-up meetings are attended only by programmers.*

***Option 2:****Correct. Scrum teams hold short, daily Scrum meetings that are led by a Scrum Master. These meetings are used to provide feedback about the project. Meetings are open to team members and interested parties. However, only team members contribute to the discussion.*

***Option 3:****Incorrect. Scrum teams hold daily Scrum meetings that can be attended by all interested parties, where only team members contribute to the discussion. The purpose of the meetings is to discuss iterative development.*

***Option 4:****Incorrect. DSDM holds short, stand-up meetings that are attended only by programmers, giving them the freedom to discuss coding developments. Scrum teams hold daily meetings that are open to external parties.*

**Correct answer(s):**

1. Extreme Programming – or XP – teams hold short meetings that are attended only by programmers  
2. A Scrum Master hosts a daily meeting that's attended by all team members and interested parties

Question

Match each methodology to the manner in which it handles change, quality, and risk during iterative development. Two options will not be used.

**Options:**

1. Crystal Orange
2. XP
3. DSDM
4. Scrum
5. Crystal Clear
6. Feature Driven Development, or FDD

**Targets:**

1. Manages risk using a combination of deep and shallow iterations
2. Maintains quality by performing unit tests and may start a new iteration after a change in requirements has been introduced
3. Uses MoSCow priorities to manage change and maintains quality by holding formal reviews during every iteration
4. Monitors quality by performing daily smoke tests but doesn't accommodate change during iterations

Answer

*Shallow iterations are recommended by Crystal Orange as a method of managing risk, in addition to deep iterations.*

*XP performs in-depth testing and conducts automatic unit tests. XP also may manage modifications to an iteration by starting a new iteration.*

*DSDM implements MoSCow priorities to manage change, and revises any modifications during the iteration's objectives setting meetings.*

*Scrum teams monitor quality by performing daily build and smoke tests. In addition, Scrum doesn't allow for change during an iteration.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

**3. Task 2: Monitoring agile teams**

It's important to monitor and report progress when working on an agile project. You can do this using a combination of metrics and reviews.

Question

You want to visually depict the amount of work remaining at the start of each iteration in a project.  
  
Which type of tool should you use to do this?

**Options:**

1. Release burndown chart
2. Parking-lot chart
3. Unit tests run

Answer

***Option 1:****Correct. Release burndown charts enable you to measure progress by determining the amount of work that remains at the beginning of each iteration in a project.*

***Option 2:****Incorrect. Parking-lot charts don't indicate the amount of work remaining at the start of each iteration. They show the progress of themes in a project by indicating the number of stories a theme contains, how many ideal days are allocated to the stories, and the percentage of the stories that are complete.*

***Option 3:****Incorrect. The unit tests run metric indicates how many tests have been run during the last build of each day. It's useful for encouraging testing, but doesn't indicate whether a feature is ready for release.*

**Correct answer(s):**

1. Release burndown chart

Question

Which tool can be used to show how feature sets are progressing?

**Options:**

1. A chart tracking milestones
2. Burn up chart
3. Release burndown chart
4. Parking-lot chart

Answer

***Option 1:****Correct. Feature-Driven Development, or FDD, uses defined milestones to measure progress in developing feature sets.*

***Option 2:****Incorrect. Burn up charts illustrate the amount of work completed and the amount of work remaining in each iteration. They do not look specifically at the progress of feature sets.*

***Option 3:****Incorrect. Release burndown charts don't use feature sets to depict progress. Instead, they use the amount of work remaining at the start of each iteration to track progress.*

***Option 4:****Incorrect. Parking-lot charts illustrate the progress that has been made on a project by tracking themes, not feature sets.*

**Correct answer(s):**

1. A chart tracking milestones

Question

Which tool or metric is the most useful for determining whether to release a feature?

**Options:**

1. Acceptance tests passed
2. A chart tracking milestones
3. A parking-lot chart
4. Unit tests run

Answer

***Option 1:****Correct. The number of acceptance tests passed at the end of each iteration can indicate the readiness of a feature for release.*

***Option 2:****Incorrect. Tracking development milestones may tell you whether development of a feature is nearing completion, but doesn't indicate whether the feature is ready for release.*

***Option 3:****Incorrect. Parking-lot charts don't indicate whether a feature is ready to be released. Instead they illustrate progress in developing the themes in a project.*

***Option 4:****Incorrect. The unit tests run metric indicates how many tests have been run during the last build of each day. It's useful for encouraging testing, but doesn't indicate whether a feature is ready for release.*

**Correct answer(s):**

1. Acceptance tests passed

Question

Your team uses the Scrum methodology. You've just completed a sprint and are about to begin the next one. Before you do so, you'd like the team to communicate about how the sprint went, and how to improve the next one.  
  
Which type of review should you use?

**Options:**

1. Sprint review
2. Sprint retrospective
3. Reflection workshop

Answer

***Option 1:****Incorrect. In sprint reviews, the team presents completed features, the sprint goal, and the product backlog commitment to there stakeholders. The stakeholders then provide comments about functionalities and any changes that they require.*

***Option 2:****Correct. Sprint retrospectives are used to obtain feedback from the team about what worked well in the previous iteration, and what needs to be improved in the next iteration. The issues that are raised are then discussed, and solutions are identified.*

***Option 3:****Incorrect. Reflection workshops are held at the end of an entire project. They are used to define what worked, what didn't work, and how issues can be improved. If you want to discuss a single sprint, you should use a sprint retrospective.*

**Correct answer(s):**

2. Sprint retrospective

Question

How should reflection workshops be conducted?

**Options:**

1. A table listing successes, problems and solutions should be used
2. Facilitators must begin with a discussion of problems
3. Good practices, issues, and solutions should be identified
4. Only the project team should be involved

Answer

***Option 1:****Correct. Most reflection workshops use a flip-chart with a table that stimulates the discussion, and is used to record the results of the meeting. This diagram should contain three sections: Keep These, Problems, and Try These.*

***Option 2:****Incorrect. The facilitator should begin the workshop on a positive note by discussing what worked well in a project.*

***Option 3:****Correct. The aim of reflection workshops is to identify good practices, as well as issues and possible solutions.*

***Option 4:****Incorrect. Reflection workshops involve the project team, as well as the customer and any other people who might have input on the project.*

**Correct answer(s):**

1. A table listing successes, problems and solutions should be used  
3. Good practices, issues, and solutions should be identified

**4. Task 3: Working with documentation**

Agile methodologies differ in their requirements for project documentation. However, they share a common approach to the role of documentation in a project.

Question

Which functions does documentation serve in agile projects?

**Options:**

1. Provides updates to off-site customers and team members
2. Records important tasks
3. Provides detailed assessments of project estimates
4. Used to obtain stakeholder approval before implementing changes

Answer

***Option 1:****Correct. Documentation is used to provide a summary of a project to customers, as well as to update off-site team members and stakeholders on project developments and progress.*

***Option 2:****Correct. Agile documents are created only when necessary and typically record only the most important aspects of a project. This helps save time, which team members can spend developing code.*

***Option 3:****Incorrect. Agile documents shouldn't contain any unnecessary detail – and only documents that are absolutely necessary should be produced.*

***Option 4:****Incorrect. Typically, traditional methods specify that changes to a project be documented and approved by stakeholders before implementation.*

**Correct answer(s):**

1. Provides updates to off-site customers and team members  
2. Records important tasks

Question

Match the different types of agile documents to their descriptions. One option will not be used.

**Options:**

1. Design documents
2. Rationale and structure documents
3. Mission and requirement documents
4. User documentation
5. Contract model documents

**Targets:**

1. Documentation that may contain sequence diagrams to expand on the system model, or specific design documents for the project being developed.
2. Short documents that outline the reasons why a system is to be developed and its overall structure.
3. Documentation that may consist only of a project metaphor or vision statement, or that may contain a project data sheet, or a product specification outline.
4. Documentation that can leverage user stories to provide end-users with information about how to use the system.

Answer

*Agile methodologies vary in their specification for design documents. XP calls for a very informal approach to design with model or design storming within iteration planning and development. FDD requires sequence diagrams to expand on the system model, and DSDM requires a number of specific design documents.*

*Agile teams often maintain a short document outlining the rationale of the system being developed and its overall structure. These documents can also serve as a design document for the project.*

*Agile methodologies recommend an incremental approach to documenting requirements, with XP recommending a short mission statement with index cards used to record features or user stories during planning. FDD requires a model of the system at project initiation while ASD recommends producing the project vision, the project data (or overview) sheet and the product specification outline.*

*In an agile environment, user stories can be leveraged to produce task-oriented and focused user documentation in an incremental manner at the end of each release cycle.*

**Correct answer(s):**

Target 1 = Option A

Target 2 = Option B

Target 3 = Option C

Target 4 = Option D

Question

What document type may simply define the mission statement of a project, or may contain a product specification outline for the project?

**Options:**

1. Mission and requirement documents
2. Design documents
3. Rationale and structure documents
4. User documentation

Answer

***Option 1:****Correct. Mission statements or requirements documents may simply contain a short mission or vision statement , or may include a project data sheet and the product specification outline.*

***Option 2:****Incorrect. Design documents may contain sequence diagrams to expand on the system model, or specific design documents for the project being developed. These documents that may range from simple informal models to detailed design and sequence diagrams.*

***Option 3:****Incorrect. These are short documents that outline the reasons why a system is to be developed and its overall structure.*

***Option 4:****Incorrect. In an agile environment, user stories can be leveraged to produce task-oriented and focused user documents in an incremental manner at the end of each release cycle.*

**Correct answer(s):**

1. Mission and requirement documents

Question

Identify some best practices to follow when creating agile documentation.

**Options:**

1. Prioritize documentation in the same manner as requirements
2. Focus documents on customer requirements
3. Ensure that documents are easily available to interested parties
4. Create most documentation at the start of a project
5. Create documents that provide accurate information about all tasks

Answer

***Option 1:****Correct. When creating documentation, you should prioritize the task in the same manner as other requirements. This includes estimating the time and effort required to develop a document.*

***Option 2:****Correct. Useful agile documents focus on the customer's needs and fulfills their requirements.*

***Option 3:****Correct. You should ensure that all agile documents are readily available to interested parties, which in turn ensures that the documents are frequently used.*

***Option 4:****Incorrect. Ideally you should create documentation towards the end of a project so that you document only what has been done.*

***Option 5:****Incorrect. Agile documents typically focus on only the most important aspects of a project. They're not expected to be detailed and it's understood that they may contain flaws.*

**Correct answer(s):**

1. Prioritize documentation in the same manner as requirements  
2. Focus documents on customer requirements  
3. Ensure that documents are easily available to interested parties

Ways to manage agile projects and monitor performance, and principles for agile documentation, have been identified.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#top)

Implementing Agile Project Scaling

Learning Objectives

*After completing this topic, you should be able to*

* *identify issues with scaling agile projects*
* *identify the features of the scrum of scrums*

**1. Issues with scaling agile projects**

It can be a challenge to scale agile methodologies for larger projects because most methodologies have been designed primarily for small teams.

It could actually be argued that scaling is contrary to the essence of agile development. This is because agile development focuses on concepts like face-to-face communication, co-ownership of code, and self-organization – which rely on small teams of developers working closely with one another.

However, the nature of a project may demand a higher number of developers. So, a development organization may be faced with the challenge of adapting an agile methodology for distributed teams.

Some aspects of agile development are easier to scale than others. For instance, developer-centric practices like pair programming can be used as easily in large teams as in small ones.

An agile communication infrastructure isn't as easily scaled. For example, the daily stand-up meetings of all team members called for in Extreme Programming – or XP – are likely to become unmanageable with too large a team.

If a team is divided into smaller units, communication and co-ordination between teams become crucial issues – or essential system attributes may be overlooked.

Some agile methodologies are more scalable than others. These include

**the Crystal family of methodologies and**

The Crystal family has been designed to cater for scaling up to approximately 100 team members. These methodologies increase in formality and capability in relation to team numbers and project criticality. In the mid-range, Crystal Orange can support up to 40 team members and includes inter-team specifications to determine how teams will interface.

**the Scrum methodology**

The Scrum methodology can be adapted for use in large projects, with several teams each following the processes it outlines while also coordinating their efforts.

A large development team can combine a methodology that isn't easily scaled with one that is. For example, a team may opt to combine XP and Scrum.

Because work in a scaled project is divided among multiple teams, it can become very complicated to manage. This requires organization and planning because, if development is to remain agile, synchronization and communication among the teams must be maintained at all times.

For example, teams should use a common basis for estimating tasks and features. Also, where teams have complex dependencies, it may be necessary to plan two or three iterations ahead.

In Scrum projects, the infrastructure that supports scaling will be developed and tested during sprints.

Graphic

*There are four phases in the Scrum process – the Product backlog phase, the Sprint backlog phase, the Sprint phase, and the Working Increment phase.*

As a result, a project starts with only one team until all the scaling requirements are in place. After that, multiple teams can be added as the project proceeds.

A recommendation is for a member of the original team to be included in each new team to advise on the scaling infrastructure.

The infrastructure requirements for scaling are included and prioritized in the product backlog, so that scaling is facilitated as soon as possible.

The phase in which scaling tasks are defined and prioritized is known as staging. The staging phase occurs during the first sprint planning meeting.

In a scaled Scrum project, ongoing communication and co-ordination of tasks occurs daily, through both

**Scrums and**

Scrum meetings are short daily meetings of each team. They are a way for the team members to "touch base" and update one another.

**Scrum of Scrums**

A Scrum of Scrums is a meeting between representatives of each Scrum team, for the purpose of coordinating all the teams' efforts.

Question

Which issues can make it difficult to scale agile development methodologies?

**Options:**

1. Agility depends on continuous collaboration among team members
2. Teams need to be added before the scaling infrastructure has been created
3. Synchronization and communication are paramount because of work being divided
4. Developer-centric practices like pair programming are difficult to apply in large teams

Answer

***Option 1:****Correct. Agile development prioritizes face-to-face communication and stand-up meetings, which are difficult to implement with a larger team.*

***Option 2:****Incorrect. It is important to have a scaling infrastructure in place before more developers are added to a project.*

***Option 3:****Correct. Synchronizing multiple teams is the most important objective of scaling agile projects, and failure to do so properly could produce significant problems.*

***Option 4:****Incorrect. Developer-centric practices like pair programming can be easily applied in both small and large teams.*

**Correct answer(s):**

1. Agility depends on continuous collaboration among team members  
3. Synchronization and communication are paramount because of work being divided

**2. The Scrum of Scrums**

The Scrum of Scrums concept is a unique way in which to facilitate the scaling of Scrum projects.

Say a project includes three teams. Each team has a daily Scrum meeting.

A nominated representative from each of the three teams then attends a second meeting, in which the activities of all the teams are discussed and coordinated. The second meeting is a Scrum of Scrums.

In smaller projects, some may prefer a more self-organized approach. In this case, each team could identify the other teams with which it needs to liaise and send a representative to the daily Scrum meetings of those teams.

Holding daily Scrum of Scrums meetings ensures ongoing communication and synchronization among all project teams.

This type of meeting is also useful for resource allocation in the beginning stages of a project, when resources frequently need to be shared by several teams.

**Drill Down Home Page**

The development team should plan the format that Scrum of Scrums meetings will take.

**Page 1 of 1: Frequency**

While scrum meetings are traditionally held on a daily basis, you may find that Scrum of Scrums meetings can be held less frequently.

For example, it may be adequate to hold a Scrum of Scrums only two or three times a week.

The optimal frequency of Scrum of Scrums meetings will depend on the size and nature of the project. This should be planned ahead.

**Page 1 of 2: Duration**

The duration of Scrum of Scrums meetings will depend on your preferences.

However, in keeping with agile principles, Scrum of Scrums meetings shouldn't be too long or waste development time unnecessarily.

**Page 2 of 2: Duration**

One approach is to timebox the meetings to, for example, 15 or 30 minutes each. This way, members are encouraged to state their issues concisely and to prioritize discussions.

Another approach is to allow the meetings to go on for as long as they have to. This would be more appropriate if you're holding less frequent meetings. Problems are solved as and when they arise, instead of being postponed for later.

Any guidelines about the duration of Scrum of Scrums meetings should be established at the outset of a project.

**Page 1 of 4: Agenda**

The traditional agenda for a daily Scrum is for each member to answer three set questions:

* What did you do yesterday?
* What will you do today?
* Do you have any roadblocks – that is, anything getting in the way of what you're doing or slowing you down?

**Page 2 of 4: Agenda**

At the Scrum of Scrums, these three questions are slightly adapted and a fourth question is added. Each attendee is asked

* What has your team done since we last met?
* What will your team do until the next time we meet?
* Is there anything getting in the way of what your team is doing, or slowing you down?
* Is your team about to do anything that might get in the way of another team?

**Page 3 of 4: Agenda**

The first part of the meeting should consist of each attendee answering the four questions.

The fourth question in the agenda is good for preventing or preparing for potentially disruptive events.

Graphic

*The fourth question, again, is "Is your team about to do anything that might get in the way of another team?"*

To keep the meetings short, it's important to keep everyone from discussing the activities of individual team members. Instead, attendees should be asked to discuss the activities of their teams as a whole.

**Page 4 of 4: Agenda**

Most teams maintain a backlog of issues to address in Scrum of Scrums meetings. This could take the form of a wiki or spreadsheet.

It's unusual for iteration planning and iteration reviews to be conducted during Scrum of Scrums meetings. If this is done at all, it's done in a much more general and informal way than in daily Scrum meetings.

**Page 1 of 3: Attendees**

In a daily Scrum, the attendees include all the members of a single team. Once the meeting is over, the team reports any obstacles to its progress to a Scrum Master.

It's important that during this meeting, team members don't feel like they're being monitored by someone in a managerial or higher authority role.

**Page 2 of 3: Attendees**

The attendees of the Scrum of Scrums, on the other hand, should be chosen by each team, based on whatever factors it considers relevant. It's also up to the team to decide whether the same person should attend all Scrum of Scrums meetings, or whether this should change with time.

One consideration is the stage of development a project is in. Ideally, the team member nominated to attend the Scrum of Scrums should be someone who's experienced and well-versed in the types of issues likely to arise at a given stage.

**Page 3 of 3: Attendees**

If the number of teams is relatively small, it may be possible for more than one person per team to attend each Scrum of Scrums. However, having too many attendees will defeat the purpose of the meetings, which is to facilitate face-to-face communication and collaboration.

Graphic

*If there are three teams, for example, the Scrum of Scrums could be attended by two members from each team.*

Scrum projects can be scaled up even further, in a recursive manner.

In a very large project, for example, there could be nine Scrum teams and three Scrum of Scrums teams of three people each. You could then nominate one member from each Scrum of Scrums to attend what is essentially a "Scrum of Scrums of Scrums."

Question

Which are features of Scrum of Scrums meetings?

**Options:**

1. They're highly focused, typically starting with each participant answering four questions
2. They must be held daily throughout a project
3. They include formal iteration planning and reviews
4. They're attended by a nominated team member from each team

Answer

***Option 1:****Correct. Typically, a Scrum of Scrums meeting starts with each team representative answering these four questions – "What has your team done since we last met?", "What will your team do until the next time we meet?", "Does your team have any roadblocks?", and "Is your team about to do anything that might get in the way of another team?"*

***Option 2:****Incorrect. Scrum meetings are expected to occur daily, but it may be sufficient to hold Scrum of Scrums meetings less often – for example, two or three times a week.*

***Option 3:****Incorrect. Formal iteration planning and reviews don't typically occur in Scrum of Scrums meetings – this is handled in each team's daily Scrum.*

***Option 4:****Correct. Each team appoints a representative, who represents the team in a Scrum of Scrums.*

**Correct answer(s):**

1. They're highly focused, typically starting with each participant answering four questions  
4. They're attended by a nominated team member from each team

**3. Summary**

There are several issues involved in scaling agile development projects. These include difficulties with adapting agile communication infrastructures to scaled projects, as well as with ensuring effective synchronization of multiple teams. Some of the Crystal methodologies and the Scrum methodology facilitate scaling. These can be combined with other methodologies such as XP in a scaled environment.  
  
In the Scrum methodology, the Scrum of Scrums concept – in which each team appoints a representative and the representatives meet frequently to collaborate – makes scaling easier.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_agsd_a03_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited