AGILE PROJECT MANAGEMENT

Agile Project Management - Complete Study Guide

Liam Smith · Version 1.0 · 4 November 2026

Instructor: **Robert Clines**. Sources: the four recordings dated **October 21, 26, 28 and November 2, 2026**, and *Introduction to Agile Project Management*, version 8.0, the supplied **187-page workbook** including its slides and appendices. This synthesis covers the supplied course sessions and Modules 1-7; it is student-prepared course material, not an official certification exam specification.

Use the Quick Review for recall, Practice Questions for application, and Source and Review Index for source navigation and qualifications.

Page key

Workbook references use **PDF-viewer page numbers, with the cover as page 1**. Lecture times are recording-relative; segment numbers are the app's analysis segments. A cited interval identifies a passage to consult; it does not mean every sentence in it supports every nearby point.

1. What to understand first

Agile organizes work so the team can learn from feedback and deliver useful pieces while needs evolve. Predictive work emphasizes coordinated planning and sequencing when the result and dependencies are sufficiently understood. Both still require requirements, estimates, quality, communication and decisions about change. Neither name repairs a poorly understood business problem.

Instructor study guidance: on November 2, Clines highlighted Modules **3, 4 and 5**, the structure of a sprint, the purpose and participants of its ceremonies, and the Product Owner, Scrum Master and team roles. He also pointed to the later modules. His recollection of certification questions was explicitly tentative and based on an exam taken years earlier. Use the emphasis to prioritize study, not as a promise about a current exam. *(Nov 2, 00:55:35-00:58:05; segments 199-207.)*

Start by explaining these distinctions without notes: predictive/adaptive/hybrid; iterative/incremental; epic/story/task; need/requirement/acceptance criterion; Product Owner/Scrum Master/team; product backlog/sprint backlog; review/retrospective; points/hours; planned/completed work; production-ready/released.

2. Module 2: approaches, feedback and frameworks

Choose the approach from the work

Approach	Main organizing idea	Indicators to investigate
Predictive / plan-driven	Plan and coordinate a sufficiently understood outcome and its sequence.	Stable requirements, costly physical rework, strong dependencies or tightly coordinated implementation.
Adaptive / change-driven	Learn, plan and deliver in smaller pieces, repeatedly using feedback.	Evolving needs, uncertainty about the solution, and opportunities for usable increments.
Hybrid	Deliberately combine approaches for different needs.	A program may have iterative software delivery and a predictive infrastructure installation.

The course uses construction, coordinated migrations and fixed-date implementations as predictive examples, and evolving websites or mobile apps as adaptive examples. Treat them as reasoning exercises. A deadline alone does not decide the method: the October 21 lecture also explicitly allows Agile projects with hard end dates. Identify what is fixed, what can flex, what can be delivered separately, and how feedback will influence the next decision. *(Oct 21, 00:35:15-00:47:24; segments 156-185.) (Oct 21, 00:53:53-01:00:24; segments 202-219.) (Oct 26, 02:26:11-02:27:40; segments 403-405.)* (Workbook, PDF pp. 17-23 and 89-90.)

For a fixed date and budget, discuss the scope that can realistically fit. For a required scope and date, examine resources, feasibility and risk. **DSDM**, as presented here, fixes cost and schedule and lets scope flex through timeboxing. A constraint tradeoff is a decision, not permission to sacrifice necessary quality. *(Oct 21, 00:57:42-01:02:00; segments 212-223.)* (Workbook, PDF p. 23.)

Iterative and incremental are related, but different

Iterative work revisits a solution using feedback to improve its correctness or suitability. **Incremental** work adds finished, tested, usable portions. Agile in the workbook combines the two: build a small useful result, learn from it, then improve or extend it. A prototype supports learning but is not automatically a production-ready increment. *(Oct 21, 00:24:58-00:32:36; segments 133-149.) (Oct 21, 00:48:50-00:49:30; segments 189-191.)* (Workbook, PDF pp. 20 and 22.)

The class's number-guessing activity illustrates the value of feedback between decisions. It does not prove that all predictive projects lack feedback. Likewise, a long interval between game releases does not establish the development method; the useful question is how work was planned, tested, adapted and integrated. Recipe development can be iterative while producing one known recipe at an event can involve a planned sequence. *(Oct 21, 01:38:30-01:45:04; segments 338-355.)*

The four Agile Manifesto preferences

Prefer	Over	Study implication
Individuals and interactions	Processes and tools	Tools support communication; they do not create shared understanding by themselves.
Working software	Comprehensive documentation	Deliver working value and produce the documentation needed to use, support and govern it.
Customer collaboration	Contract negotiation	Work with stakeholders to understand and refine the result.
Responding to change	Following a plan	Use plans and revise them as evidence changes.

The right-hand items still have value. These are preferences, not bans on processes, documents, contracts or planning. The course extends the working-software idea to working solutions beyond software. *(Oct 21, 00:50:41-00:53:19; segments 194-198.) (Nov 2, 00:01:53-00:05:32; segments 5-16.)* (Workbook, PDF pp. 21-22 and 155.)

Framework recognition

Framework	Recognize it by	Boundary to remember
Scrum	Sprints, a prioritized backlog, distinct roles, recurring collaboration and inspection.	Scrum is one Agile framework, not a synonym for all Agile work.
DSDM	Fixed time and cost with scope selected to fit.	Not every constraint arrangement fits this model.
Kanban	Visualizing/organizing flow, prioritizing work and limiting work in progress.	A board alone does not remove bottlenecks or manage priorities.
Extreme Programming (XP)	Pairing, continuous testing, test-based feedback and frequent change.	The workbook's strict co-location/minimal-documentation characterization is its course description; avoid making it a universal rule.
Feature-Driven Development (FDD)	Models inform a feature list, then feature planning, design and build in small portions.	Features still need a connection to business needs.

(Oct 21, 01:00:26-01:10:42; segments 220-245.) (Workbook, PDF pp. 23-25 and 40-42.) Historical dates and the classification of Spiral/RAD are lower-priority course background. This guide does not turn the workbook's disputed historical labels or its claim about Scrum's ownership into memorization targets. See the source index.

3. Module 2: Scrum roles and the delivery loop

Who decides what?

Role	Main responsibility in the course	Avoid confusing it with
Product Owner (PO)	Represents business needs, clarifies requirements, orders the product backlog and makes value/priority decisions with stakeholders.	Assigning more sprint work than the team can accomplish, or assuming the PO must be a BA.
Scrum Master (SM)	Coaches and facilitates the process, helps remove impediments and protects the team's ability to focus.	A task boss who dictates every implementation decision.
Team members	Bring the skills needed to deliver, estimate/select feasible sprint work, organize its execution and coordinate quality.	Individuals waiting passively to be told what to do.
Project manager / external coordination	Where the organization uses this role, connects budgets, approvals, dependencies, reporting and release obligations with the team's work.	A mandatory additional Scrum role or the same accountability as Scrum Master.

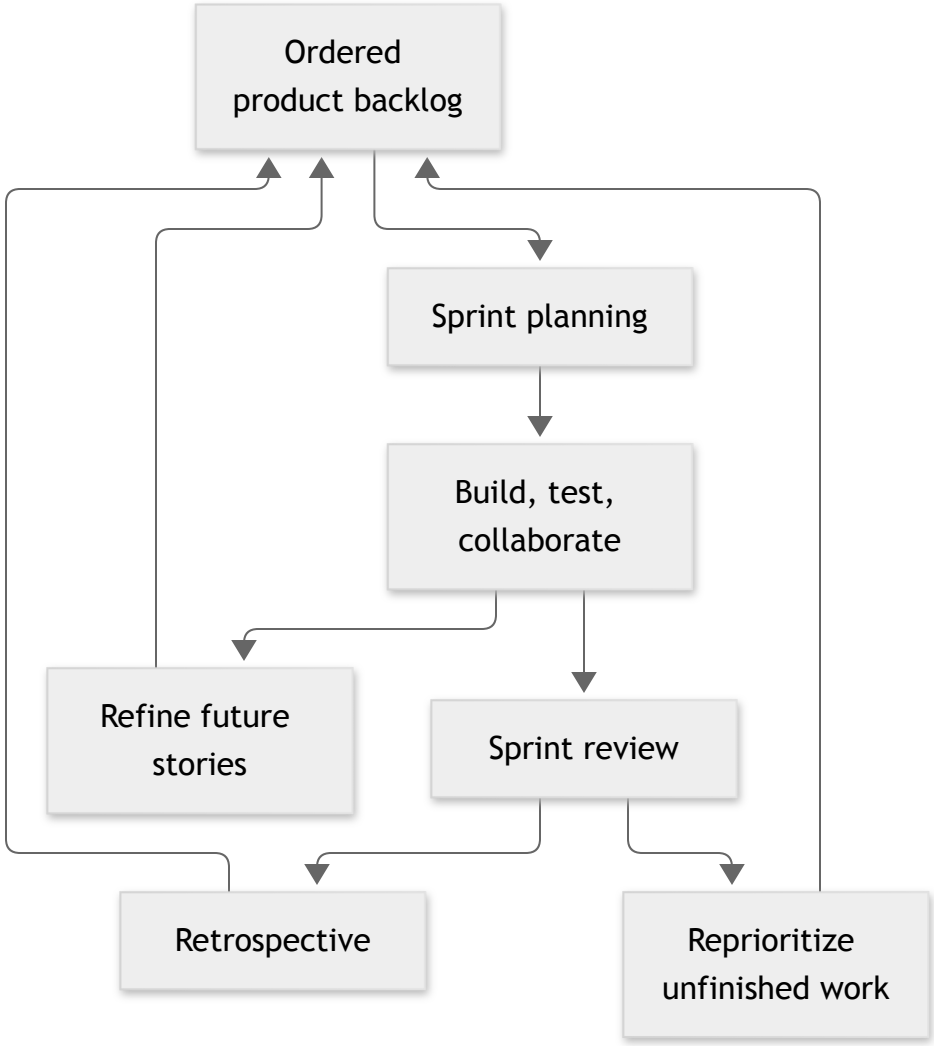
A BA may be a good Product Owner because of elicitation and business communication skills; the lecture explicitly says a Product Owner does **not** have to be a BA. Authority to decide priorities matters more than the job title. Self-management does not remove product priorities, organizational obligations or accountability. *(Oct 21, 01:16:49-01:25:59; segments 258-278.) (Oct 26, 00:21:13-00:28:29; segments 56-71.) (Oct 28, 01:10:58-01:11:58; segments 186-188.)* (Workbook, PDF pp. 27-29 and 87-88.)

The course recommends a small, cross-functional team, usually **nine or fewer team members if possible**. This is the instructor's course convention; the workbook has inconsistent size wording. Do not use that number as a universal Scrum standard or infer that every role must be a separate employee. *(Nov 2, 00:26:44-00:30:20; segments 79-90.)* (Workbook, PDF pp. 28 and 160.)

Product backlog versus sprint backlog

The **product backlog** holds potential product work in priority order and evolves as the business learns. The **sprint backlog** is the work the team selects for the current sprint, with its execution details. Product priority comes from the PO working with the business; feasible amount and execution are team decisions. *(Oct 21, 01:10:45-01:16:46; segments 246-257.)* (Workbook, PDF pp. 26-27 and 81.)

At sprint end, incomplete work returns to the product backlog for reassessment. It receives no completed-story credit and does not automatically become the highest-priority item in the next sprint. Previous effort is not sufficient reason to finish a now-unwanted feature. The PO may still rank it first if its remaining value justifies that choice. *(Oct 26, 02:09:13-02:10:51; segments 364-367.) (Nov 2, 00:25:02-00:25:34; segment 75.)*



The diagram summarizes the course loop. Daily collaboration happens during execution; refinement prepares future work and does not automatically commit it. A release can contain the results of several sprints. (Workbook, PDF pp. 27, 80–84 and 130.)

---

4. Module 3: requirements, epics and user stories

Elicit needs rather than merely collect requests

A **need** identifies a problem or desired benefit. A **requirement** states a capability or condition that would meet it, with enough specificity for its intended use. Elicitation means collaborating, asking questions, checking interpretations and returning for feedback. Simply recording initial wishes and disappearing until delivery can preserve misunderstandings. Interviews, brainstorming and affinity diagramming are course examples; choose techniques that help expose gaps and differing perspectives. *(Oct 26, 00:00:49–00:06:53; segments 3–15.) (Oct 26, 01:23:46–01:27:22; segments 263–277.)* (Workbook, PDF pp. 49–51.)

Requirement category	What it expresses	Editorial example for a study app
Business	The organizational goal or reason for change.	Improve access to course revision materials.
Stakeholder / end-user	What a particular person or group needs.	A student needs to identify weak subject areas.
Solution - functional	Behavior or capability of the solution.	Record a submitted answer and show feedback.
Solution - non-functional	A quality, performance or operating condition.	Meet an agreed response-time target under a defined load.
Transition	A temporary capability needed to move to the new state.	Migrate old study records and train support users before retiring the old system.

There are **four top-level categories**; functional and non-functional are subdivisions of solution requirements. An ongoing product behavior is not a transition requirement merely because it changes an application's state. The banking discussion uses security and dark mode as non-functional examples; classify the actual statement carefully rather than assuming every security-related behavior belongs in that category. *(Oct 26, 00:22:35-00:26:51; segments 59-67.) (Oct 26, 00:34:12-00:42:34; segments 97-126.)* (Workbook, PDF pp. 51 and 61.)

From epic to estimable story

An **epic** is a large story that needs decomposition before the team can estimate and deliver its pieces. “Manage my money” is too broad: view a balance, search a transaction and pay a bill are different capabilities. A small story that is difficult to estimate may need investigation or clarification; uncertainty alone does not prove it is a large epic. The course's “can we estimate it?” test is a useful prompt to refine, not a substitute for judging size and clarity. *(Oct 26, 00:43:02-00:50:49; segments 130-150.) (Oct 26, 00:55:36-01:06:26; segments 171-206.)* (Workbook, PDF pp. 51-53.)

Use the pattern **As a [user role], I want to [capability], so that [benefit]**. Preserve the intended user and value rather than obsessing over exact template words. The class allows an implicit benefit when it adds nothing useful. A story says what is needed and opens a conversation; it is not a complete technical specification, a task list or a promise that every requested feature will be delivered. The course uses stories in place of formal functional-requirements documents, but this does not eliminate required documentation in every organization. *(Oct 26, 00:46:20-00:50:49; segments 141-150.) (Oct 26, 01:08:41-01:19:01; segments 212-240.)* (Workbook, PDF pp. 52-55.)

The three Cs and acceptance criteria

C	Purpose	Example question
Card	Briefly capture a user need and benefit.	Who needs to do what, and why?

C	Purpose	Example question
Conversation	Develop shared understanding and expose assumptions.	Which users, exceptions, inputs and constraints matter?
Confirmation	Agree how the result will be checked.	What observable result would demonstrate acceptance?

Acceptance criteria express the conditions a particular story must meet and guide testing. They can evolve as analysis improves understanding. Interpret the lecture's "how it will be achieved" wording as making success concrete, rather than requiring every criterion to prescribe a technical design. "Easy to use" is vague; a specific authorized user completing a specified action and receiving a defined result is checkable. *(Oct 26, 00:51:23-00:53:06; segments 154-156.) (Oct 26, 01:06:28-01:16:01; segments 207-233.)* (Workbook, PDF pp. 52-55.)

Editorial example: As a student, I want to filter practice questions by module so I can focus on a weak subject. Ask which modules, how mixed questions are classified and what an empty result should show. Possible criteria: choosing Module 5 displays only its tagged questions; removing the filter restores the full set; a module with no questions shows an informative empty state. These criteria confirm behavior without dictating the code structure.

Deliver a useful slice

An increment should provide usable functionality across the layers needed for that result. Building every database table in one sprint and every screen in the next may produce no usable feature in the first sprint. A bounded account-viewing capability can include its relevant interface, logic, data and tests together. Identify real users with different goals and proficiency; one generic user perspective can hide requirements. *(Oct 26, 01:17:53-01:20:48; segments 239-244.)* (Workbook, PDF pp. 55-56.)

---

5. Module 4: planning, events and progress

Five planning levels in the course

Planning level	Purpose, people and output
Vision	Business leadership and product stakeholders identify why the work matters and its high-level goal; this provides guiding direction.
Roadmap	PO and business, with team input as needed, describe the strategic sequence for rolling out capabilities.
Release	PO and team consider capabilities across several sprints; output is a prioritized, relatively estimated candidate backlog and tentative allocation.
Sprint	PO clarifies priorities; team evaluates capacity, selects feasible work for the next sprint and develops its sprint backlog.

Planning level	Purpose, people and output
Daily	Team coordinates near-term work and impediments, with SM facilitation in the course model; output is updated understanding and follow-up actions.

The workbook also calls the fifth level a status meeting. Keep that course wording in context: the useful activity is brief team coordination, not reporting to a manager for permission to act. Release planning is a forecast; assigning a story to a future sprint does not guarantee it will be selected. Refinement helps prepare candidate work and questions, rather than replacing sprint planning. *(Oct 26, 01:50:55-02:04:30; segments 319-351.)* (Workbook, PDF pp. 78-82.)

Know the purpose and people for each activity

Activity	What it is for	Who is involved in the course model
Sprint planning	Understand candidate stories, estimate, account for capacity and agree feasible sprint work.	PO and team; SM supports the process.
Daily stand-up	Coordinate work since the last meeting, intended next work and impediments.	Team; SM facilitates in the taught format.
Backlog refinement / story time	Clarify upcoming stories, dependencies and questions before later selection.	PO and relevant team members; stakeholders supply answers as needed.
Sprint review	Inspect and demonstrate completed functionality with the business; obtain feedback.	Team, PO and relevant stakeholders.
Sprint retrospective	Improve how the team works; preserve successes and address problems.	Team with SM facilitation in the course example.
Release planning	Look across several sprints and coordinate an upcoming release.	PO and team, connected to organizational release needs.

These are the course's activities, including refinement and release planning; this table is not a claim that all six are formal events required by a current Scrum standard. *(Oct 26, 02:00:02-02:13:23; segments 340-374.) (Oct 28, 01:19:57-01:22:15; segments 207-212.) (Nov 2, 00:55:35-00:58:05; segments 199-207.)* (Workbook, PDF pp. 79-84 and 130.)

The course's daily stand-up is **15 minutes or less**, held consistently, using three prompts: work since the previous meeting, intended next work, and impediments. Raise an obstacle and arrange follow-up; do not consume the meeting with detailed design or problem solving. The three prompts and facilitation arrangement are the taught format. They are not presented here as the only possible form of daily collaboration. *(Oct 26, 00:13:12-00:14:50; segments 33-36.) (Oct 26, 02:01:56-02:03:55; segments 345-350.)* (Workbook, PDF pp. 81-82.)

Reviews concern the **product/result**; retrospectives concern the **way the team works**. In the course case, stakeholders see work during the sprint so the review is not their first exposure. The retrospective should identify an action and preserve a useful practice, with no finger-pointing. *(Oct 26, 02:09:13-02:11:53; segments 364-*

370.) (Oct 28, 01:16:29-01:22:15; segments 199-212.) (Workbook, PDF pp. 83 and 129-130.)

Cadence and change during a sprint

The lecture/workbook teach **one-to-four-week sprints**, with **two to three weeks** an instructor preference; November 2 clarifies the garbled October 26 minimum-duration exchange. The illustrative two-week schedule starts with planning, has daily collaboration, includes look-ahead refinement and ends with review followed by retrospective. These sample timings are not fixed calendar requirements. *(Oct 26, 02:11:56-02:13:23; segments 371-374.) (Nov 2, 00:21:18-00:24:24; segments 58-73.)* (Workbook, PDF pp. 25, 84 and 158.)

New requests should go through the PO and product backlog rather than an executive directly redirecting the team. Discuss urgency, value, capacity and what would change. In the course's exceptional termination scenario, only the **PO** terminates the sprint, then the backlog is reprioritized and work replanned. Do not turn every new request into an automatic cancellation rule. *(Oct 26, 02:04:30-02:09:09; segments 352-363.)* (Workbook, PDF p. 82.)

Task board, burndown and dependencies

A task board makes current sprint work visible, commonly as **to do, in progress, done**. The product backlog and longer release plan serve different horizons. A burndown charts **remaining work** against elapsed sprint days; in this course that work is measured in **hours**. The ideal line reaches zero at the end. A consistently high actual line invites investigation; it does not prove a particular cause or an individual's poor performance. *(Oct 26, 02:14:37-02:18:36; segments 378-386.)* (Workbook, PDF pp. 85-86.)

Editorial calculation: with 80 hours remaining across 10 workdays, a straight ideal line burns 8 hours per day. At the end of day 4, ideal remaining work is 80 - 4 × 8 = **48 hours**. If actual remaining work is 64 hours, it is **16 hours above ideal**. Ask about discovered work, dependencies, availability and estimation rather than declaring the team 16 hours behind by certainty. A remaining-work estimate can rise when understanding improves.

The course revisits deliverables, WBS decomposition and sequencing to connect Agile work with project planning. Identify what must be produced before deciding its sequence; a breakdown is not a dependency network. The workbook's **8/80** sizing rule is explicitly a guideline, with exceptions. Its diagram includes activities beneath deliverables; preserve that as a course illustration rather than redefining every WBS in that form. *(Oct 26, 01:42:08-01:50:54; segments 298-318.)* (Workbook, PDF pp. 75-78.)

---

6. Module 5: estimation, prioritization and velocity

Estimates improve as information improves

An estimate is a reasoned forecast with uncertainty. Communicate assumptions and revise when evidence changes. **Cost** in the course is the money charged/paid for the work; **budget** allocates the estimate against scheduled work and time. An early top-down business-case estimate does not become more accurate merely by splitting it into smaller numbers, nor should it be preserved for political convenience when better information contradicts it. *(Oct 28, 00:01:24-00:10:44; segments 4-24.)* (Workbook, PDF pp. 111-112.)

Estimate class	Workbook's illustrative range	Workbook technique association
Order of magnitude	±50-100%	Top-down
Conceptual	±15-50%	Top-down
Preliminary	±10-30%	Top-down **or** bottom-up
Definitive	±5-15%	Bottom-up
Control	±3-5%	Bottom-up

These are the workbook's suggested classes and ranges, not independently verified AACE rules or universal PMI ranges. October 28 groups preliminary with bottom-up; workbook p. 111 explicitly allows both methods. Use that distinction if comparing the sources. The tentative “61%” environment-effect statistic has no supporting source here and is excluded from graded recall.

Traditional techniques named in the course are analogous/top-down, parametric, bottom-up, expert judgment and PERT/three-point. The workbook resolves the transcript's unclear “Kurt” as PERT in this list. Their basic purposes are comparing similar work, applying a rate/relationship, aggregating detailed estimates, consulting informed expertise and considering optimistic/most-likely/pessimistic outcomes. *(Oct 28, 00:49:10-00:49:49; segments 138-145.)* (Workbook, PDF p. 112.)

Relative size is not clock time

Story points compare the relative size of stories within the team's estimation context. Consider complexity, effort, uncertainty, dependencies and the completion standard. A smaller number means a relatively smaller item; it is not a fixed duration or a command to do it first. The course later breaks selected stories into tasks and estimates task work/duration in hours for planning and reporting. Some experienced teams may rely more on points; the lecture acknowledges this. *(Oct 28, 00:10:46-00:14:00; segments 25-35.) (Oct 28, 00:18:51-00:35:06; segments 62-100.)* (Workbook, PDF pp. 113-115.)

The workbook calls its scale Fibonacci and uses endpoints 0 and 100; its sample solution also uses 20. Treat this as a **modified estimation-card scale**, not a mathematical claim that every displayed value belongs to the exact Fibonacci sequence. For practice, 1, 2, 3, 5, 8, 13 illustrates widening relative sizes; a large or uncertain item prompts discussion and possible decomposition. A “five-second last-name field” is a class illustration, not a reliable estimate for a real feature with testing and integration obligations.

Planning Poker

1. Prepare stories sufficiently for estimation; bring the relevant context and questions.
2. The PO presents a candidate and the team discusses it.
3. Each estimator privately selects a value.
4. Reveal the estimates together so early speakers do not anchor everyone else.
5. Discuss differences, especially outliers: assumptions, skill, complexity and dependencies may differ.
6. Estimate again until the team reaches an informed shared estimate.

Agreement should emerge from understanding, not pressure to match a senior person or an arithmetic average that hides disagreement. A digital tool or physical cards can preserve independent selection and simultaneous disclosure. *(Oct 28, 00:22:21-00:28:21; segments 71-84.)* (Workbook, PDF pp. 114-115.)

Prioritize for value, then test feasibility

The course's ordered considerations are **business value, risk, difficulty and size**. Smaller stories are preferred when other considerations are equal; the smallest story is not automatically more important than a high-value or dependency-enabling item. The PO works with stakeholders to rank work, while the team makes implementation and capacity constraints visible. Calling everything “highest priority” avoids the actual decision. *(Oct 28, 00:36:00-00:42:26; segments 106-120.) (Oct 28, 01:12:57-01:13:57; segments 191-192.)* (Workbook, PDF pp. 115-116.)

Velocity counts completed stories

Sprint velocity = sum of the story points for stories completed to the agreed completion standard during that sprint. Do not award partial-story points or count work merely planned. Average several comparable sprints to support forecasts. The average guides selection; it is not an exact quota, an individual productivity score or a basis for converting points across unrelated teams. *(Oct 28, 00:42:29-00:48:20; segments 121-134.)* (Workbook, PDF p. 116.)

Editorial example: a sprint selects stories worth 3, 5, 8 and 13 points. The first three finish; the 13-point story is 90% complete. Velocity is **3 + 5 + 8 = 16 points**, not 29 and not 27.7. The unfinished story returns for reprioritization. If recent completed velocities are 16, 20 and 18, their average is **18 points/sprint**. A comparable remaining backlog of 54 points suggests about **three sprints**, assuming stable scope, team availability and point scale; new information can change that forecast.

A **calibration sprint** in this workbook is a learning/dry-run period with real work, deliberately excluded from the case's formal velocity metrics. It is a course adoption technique, not a required additional Scrum event. The case illustrates over-selection and learning; its spoken counts conflict, so the exact calibration-story arithmetic is not used for grading. *(Oct 28, 00:43:43-00:46:19; segments 123-128.) (Oct 28, 01:14:00-01:18:59; segments 193-204.)*

---

7. Module 6: execute, inspect and integrate

Definition of Done and release readiness

Acceptance criteria confirm a particular story's required behavior; the **Definition of Done** establishes a shared completion standard for the work. A developer saying “my unit tests pass” may leave integration, documentation, approvals or readiness work incomplete. Discuss and budget all required work before assuming the result is finished. *(Oct 28, 00:19:38-00:21:16; segments 64-67.) (Nov 2, 00:32:45-00:35:21; segments 97-105.)* (Workbook, PDF pp. 113-114 and 159-160.)

In the course's corporate example, done means ready to move into production at the next release date: needed tests, scripts, documentation, training, approvals, communications and packaging are complete. Distinguish **production-ready** from **already deployed**. Actual release may follow a separate approved schedule.

Requirements differ by organization; this is the example's readiness checklist, not a claim that every story in every team requires every listed item. (Workbook, PDF pp. 86-88 and 160.)

The failed demonstration in the October 28 case is a lesson in testing and collaboration. Business stakeholders should see and discuss work during the sprint; waiting for the final review to discover a mismatch defeats the feedback loop. The manually reviewed claims anchored to segments 199-200 are supported by that surrounding passage and retained here with their source qualification in the index. *(Oct 28, 01:16:29-01:18:59; segments 199-204.) (Oct 28, 01:35:09-01:39:10; segments 243-252.)*

Learning from the case, not memorizing its calendar

The sample team's early problems are taking too much work, waiting for directions, overlong stand-ups, insufficient testing and too little stakeholder feedback. Later improvements include clearer stories, more realistic selection, self-directed help, better look-ahead refinement and constructive retrospectives. A successful sprint is still a reason to inspect how the result was achieved; one easy sprint does not establish a permanent capacity. *(Oct 28, 01:14:00-01:24:55; segments 193-218.) (Oct 28, 01:39:15-01:40:17; segments 253-254.)* (Workbook, PDF pp. 127-136.)

Situation	Useful response	Why it helps
A manager tells a member to change priorities mid-sprint.	Clarify urgency and assignment authority; involve SM/PO and negotiate through the agreed process.	Protects focus while making real organizational obligations visible.
An external team misses a dependency.	Surface and coordinate the dependency; reforecast and return unfinished work for PO reprioritization.	Changing method names does not remove dependency risk.
Stakeholders request more during a demo.	Capture the request, clarify value and let the PO rank it.	Feedback can inform future work without silently expanding current capacity.
A retrospective becomes blame.	Facilitate a discussion of causes, practices and agreed improvements.	Learning requires honest participation and follow-through.

(Oct 28, 01:28:39-01:36:02; segments 228-244.) (Oct 28, 01:42:51-01:45:18; segments 260-264.) (Workbook, PDF pp. 132-136.)

Teams, availability and outside work

Effective teams need communication, cross-functional help, stable work conditions, usable virtual collaboration and realistic dedication. Cross-functionality means covering necessary work collaboratively, not assuming everyone has every specialist permission or skill. Coordinate scarce experts and external approvals explicitly. *(Oct 28, 01:45:22-01:53:53; segments 265-282.) (Oct 28, 02:01:35-02:08:06; segments 304-316.)* (Workbook, PDF pp. 136-138 and 150-152.)

For shared resources, plan against **actual allocation**, not nominal headcount. Concentrated blocks of time can reduce switching and handoff problems. If the team also handles production support, the instructor suggests using recent support demand across four iterations to reserve capacity. More support than expected can leave a

planned story unfinished; less support may allow another candidate after discussion with the PO. Keep hours and points distinct. *(Oct 28, 02:08:08-02:11:55; segments 317-326.)*

Editorial capacity example: three people nominally provide 10 days × 6 project hours = 60 hours each. Two are fully allocated and one is 50% allocated: 60 + 60 + 30 = **150 project hours**. Reserving 24 hours for anticipated support leaves **126 hours** before other known absences or obligations. This is an hours-based planning assumption, not 126 story points.

Remote work needs explicit communication habits, tools, access and participation. The instructor recommends early relationship-building and face-to-face time where feasible. Camera, co-location and meeting recommendations are adaptable team practices; physical presence alone does not prove useful collaboration. Agree ground rules with the team, make them accessible and revisit them when conditions change. *(Oct 28, 01:50:05-02:03:04; segments 275-306.) (Nov 2, 00:38:12-00:40:41; segments 113-118.)*

Work inside the organization

Business cases, budgets, reporting, change approvals and release processes continue around Agile delivery. The course describes an external PM/BA arrangement for coordinating those obligations and Scrum of Scrums for cross-team dependencies. A mixed program may contain both predictive and adaptive projects. Governance should connect the work rather than disappear because a team uses a backlog. *(Oct 26, 02:19:14-02:27:40; segments 388-405.)* (Workbook, PDF pp. 87-90 and 149.)

The workbook describes **hardening sprints** for corporate release/documentation work. Treat that as its integration example: plan any such effort and capacity explicitly, and do not label unfinished required testing or documentation “done” merely because it will be completed later. Review completion/readiness expectations with the organization.

---

8. Module 7: myths, adoption and improvement

Myth	Supported study conclusion
Agile does no documentation.	Produce useful, necessary documentation and satisfy actual support/governance needs.
Agile does no planning.	Planning is distributed across horizons and repeated as knowledge changes.
Agile cannot work with distributed teams.	It needs deliberate virtual collaboration, as other approaches do.
Self-management means no discipline.	The team must organize its work and honor shared responsibilities.
Agile cannot scale.	Coordinate teams and dependencies; the course gives Scrum of Scrums and shared-backlog examples.
Agile is an easy cure for delivery problems.	Adoption needs capability, support, experimentation and attention to underlying causes.

Myth	Supported study conclusion
Waterfall is obsolete or cannot change.	It remains useful for some work and can manage change through an established process.
Agile always finishes the whole project faster.	Earlier usable increments do not establish a shorter total duration for every project.
All formal reporting is waste.	Reports may support organizational decisions beyond the team's immediate needs.

(Nov 2, 00:01:53-00:20:08; segments 5-48.) (Workbook, PDF pp. 155-157.) The lecturer's one-year comparison illustrates the same-scope assumption; it is not evidence that every Agile and predictive project takes exactly the same total time. Failure-rate anecdotes and unsupplied statistical reports are not used as numerical facts in the practice bank.

Diagnose the common adoption patterns

Watch for oversized stories, repeated carryover, very long cycles, excessive advance design, large teams, unclear done, outside interference and tiny resource allocations. Identify the mechanism instead of simply blaming “lack of Agile.” Clarify a story; expose a missing dependency; negotiate availability; improve the completion agreement; test a more suitable cadence. *(Nov 2, 00:21:18-00:35:21; segments 58-105.)* (Workbook, PDF pp. 158-160.)

The course's continuous-improvement targets are **shorter sprints where useful, smaller effective teams, greater dedication, better co-location/collaboration, team-owned ground rules, management education, and coaching/mentoring**. They are improvement directions with tradeoffs, not automatic instructions to cut every team or halve every sprint. Leadership needs realistic expectations and a clear understanding of decision boundaries. *(Nov 2, 00:35:53-00:43:52; segments 108-125.)* (Workbook, PDF pp. 160-161.)

“Fail fast” means detect a failure, learn from it and apply the lesson promptly rather than repeatedly making the same error. Experiment deliberately and review results. Scrum, Kanban, predictive and hybrid approaches can each fit different conditions; changing labels without changing behavior provides little benefit. *(Oct 26, 00:16:03-00:18:01; segments 41-45.) (Nov 2, 00:49:23-00:51:53; segments 165-169.)*

---

9. Apply the ideas to one coherent scenario

Editorial exercise: A study app must support a course exam in six weeks. Students are unsure which filter and feedback designs will help them. A separate authentication migration has a fixed cutover and approvals. Three developers share production support.

1. **Approach:** trial small usable study features with feedback, while explicitly coordinating the cutover and its dependencies. A fixed exam date makes scope/capacity tradeoffs necessary; it does not by itself decide the method.
2. **Requirements:** separate the learning-access business goal, student needs, application functions, quality targets and temporary migration work.

3. **Stories:** break “improve revision” into valuable, estimable slices, such as filtering questions by module or reviewing explanations after checking an answer.
4. **Confirmation:** agree observable pass/fail conditions and necessary quality/readiness work. Include exception and access cases.
5. **Selection:** let the PO rank value, risk, difficulty and size. Let the team estimate and select work using actual allocation, support reserve and dependencies.
6. **Execution:** coordinate briefly each day, resolve detailed issues in follow-up and show work to students before the sprint review.
7. **Inspection:** demonstrate completed work, count only completed stories for velocity and return unfinished work for reprioritization.
8. **Improvement:** choose a concrete retrospective action, such as refining the next dependency-bearing story earlier, and check whether it helped.

Explain the reasoning before trying the practice bank. The questions are original exercises based on the supplied teaching; no item is represented as an actual certification question.