JUPITER PRIME

Consulting • Training • Advisory

جوبيتر برايم للاستشارات والتدريب

COURSE BROCHURE

Advanced Agile Software Delivery Management Training Course

12 – 23 September 2027
Kuala Lumpur

jupiterprime.uk

info@jupiterprime.uk

Registered in the United Kingdom and Saudi Arabia

Advanced Agile Software Delivery Management Training Course

Ref: 786_29786 **Date:** 12 – 23 September 2027 **Location:** Kuala Lumpur **Fees:** 8500 GBP

Agile Software Delivery Management is a 2-week advanced training course for delivery managers and product leads in software firms, built around an agile delivery plan for a real software product. Software companies now ship to many clients from shared codebases, and the gap between sprint activity and predictable releases decides whether contracts are renewed. Jupiter Prime Training delivers Agile Software Delivery Management for leaders who coordinate several teams on one product.

Why Agile Software Delivery Management Matters Now

Three forces shape agile software delivery today. Clients buying software increasingly ask for incremental releases and evidence of progress rather than a single acceptance date. Product codebases are shared by several teams, so dependencies, integration cadence and technical debt now limit speed more than individual team skill. Investors and boards also expect forecasts grounded in measured throughput rather than optimism. A firm that manages these well releases predictably and keeps client trust; one that manages them badly overruns, discounts and burns out engineers. The course connects team practice to enterprise IT governance through COBIT 2019 and to service transition through ITIL 4.

Who Gains Most from Agile Software Delivery Management

- Managers accountable for release forecasts across several Scrum or Kanban teams on one product
- Product leads who own backlog priorities and negotiate scope with client stakeholders
- Engineering heads responsible for codebase health, technical debt and integration cadence
- Account and bid leads who shape statements of work and change clauses for software projects
- Agile coaches who design team topology and working agreements across a product line

Product groups that send their delivery manager, product lead and engineering head together gain most, because they return with one shared view of flow, forecasts and contract scope.

Agile Software Delivery Management Capabilities

Participants practise turning many teams' sprint and board activity into release commitments a client can rely on.

- Design a multi-team operating structure using scaled Scrum and Kanban patterns — engineering heads
- Calculate cycle time, throughput and work item age to spot flow constraints — delivery managers
- Build probabilistic release forecasts with Monte Carlo simulation — clients and sponsors
- Weigh technical debt repayment against feature demand in backlog ordering — product leads
- Specify agile contract terms for scope, change and acceptance — account leads and legal advisers
- Communicate delivery risk and progress through product-level reviews — boards and client steering groups

Key Agile Software Delivery Management Terms

- Cycle time — the elapsed time from when work on an item starts until it is finished and ready for release.
- Throughput — the number of work items a team or product group completes in a fixed period.
- Work item age — how long an unfinished item has been in progress, used as an early warning of stalled work.
- Monte Carlo forecast — a simulation that replays historical throughput many times to give release dates with stated probabilities.
- Technical debt — design or code shortcuts that speed delivery now but raise the cost of later change.
- Target cost contract — an agreement where client and supplier share savings or overruns against an agreed estimate for evolving scope.

Agile Software Delivery Management Day by Day

Day 1: Delivery Context in Software Firms

- Agile Manifesto Values Revisited — Where Product Firms Drift From Them
- Project Versus Product Funding Models — How Budgets Shape Team Stability
- Software Value Stream Map — Wait Time Between Idea And Release
- Client Portfolio Pressures — Competing Roadmaps On One Shared Codebase
- Delivery Health Baseline — Current Lead Time, Defects And Morale

Day 2: Scaling Scrum Across Teams

- Nexus Integration Team — Coordinating Several Scrum Teams On One Increment
- Large-Scale Scrum Feature Teams — Reducing Handoffs Between Component Groups
- Scaled Agile Framework Agile Release Train — Cadence For Larger Product Lines
- Scrum Of Scrums Practice — Surfacing Cross-Team Impediments Each Day
- Framework Selection Criteria — Matching Scaling Model To Product Size

Day 3: Kanban and Flow Design

- Kanban Method Practices — Visualising Work And Limiting Work In Progress
- Board Design For Multi-Team Products — Columns, Swimlanes And Blocker Tags
- Classes Of Service — Expedite, Fixed Date And Standard Work
- Service Level Expectations — Percentile Targets For Item Completion
- Upstream Kanban Discovery — Options Before Commitment To Build

Day 4: Flow Metrics and Their Signals

- Cycle Time Scatterplot — Spotting Outliers And Process Variation
- Throughput Run Chart — Stability Of Completion Rate Over Weeks
- Cumulative Flow Diagram — Bottleneck Stages And Queue Growth
- Work Item Aging Chart — Intervening Before Items Stall
- Flow Efficiency Calculation — Active Versus Waiting Time In Delivery

Day 5: Release Planning and Forecasting

- Product Roadmap Horizons — Now, Next And Later Commitments
- Story Mapping For Release Slices — Smallest Releasable Client Value
- Monte Carlo Simulation — Probabilistic Dates Instead Of Single Estimates
- Release Burn-Up Chart — Scope Change Versus Completed Work
- Dependency Board — Cross-Team Sequencing For Each Release

Day 6: Engineering Practices and Technical Debt

- Definition Of Done Across Teams — Shared Quality Bar Per Increment
- Continuous Integration And Trunk-Based Development — Integration Risk Per Release
- Technical Debt Quadrant — Deliberate Versus Accidental Shortcuts
- Debt Register And Interest Estimate — Ranking Repayment Candidates
- Capacity Allocation Policy — Feature, Debt And Defect Work Split

Day 7: Product Ownership at Scale

- Product Goal And Backlog Hierarchy — Epics, Features And Stories
- Weighted Shortest Job First — Sequencing By Cost Of Delay
- Area Product Owner Roles — Splitting Backlog Ownership Sensibly
- Multi-Client Feature Requests — Configurable Versus Bespoke Decisions
- Sprint Review With Client Stakeholders — Feedback That Changes Priorities

Day 8: Client Contracts for Agile Delivery

- Fixed Price Versus Time And Materials — Risk Allocation Per Model
- Target Cost And Capped Contracts — Shared Incentives On Evolving Scope
- Change For Free And Exit Clauses — Flexibility Without Disputes
- Incremental Acceptance Criteria — Sign-Off Per Release Slice
- Statement Of Work Drafting — Outcomes, Cadence And Governance Forums

Day 9: Governance, Risk and Service Transition

- COBIT 2019 Build And Acquire Practices — Board Oversight Of Software Change
- ITIL 4 Release Management Practice — Handover From Delivery To Operations
- Delivery Risk Register — Dependency, Staffing And Vendor Exposures
- Portfolio Kanban For Product Lines — Funding Flow Across Products
- Executive Delivery Dashboard — Flow Metrics Leaders Actually Read

Day 10: Practising the Agile Delivery Plan

- Team Topology And Scaling Choice Applied — Structure For The Chosen Product
- Flow Baseline And Forecast Applied — Release Dates With Probabilities
- Debt Allocation And Contract Terms Applied — Capacity And Client Agreement
- Peer Challenge Session — Testing Assumptions Before Client Presentation
- Agile Delivery Plan — Structure, Forecast, Contract And Risk Sections

Applying Agile Software Delivery Management After the Course

- Pull the last three months of completed items from your team boards, publish a cycle time scatterplot and throughput run chart, and agree service level expectations with each team.
- Replace the next single-date release estimate with a Monte Carlo forecast and present it to the client steering group alongside a technical debt capacity allocation.

Agile Software Delivery Management Questions FAQ

What is Agile Software Delivery Management and why does it matter now?

Agile Software Delivery Management is the discipline of coordinating several agile teams so a software product releases predictably. It matters now because clients expect incremental value, shared codebases create cross-team dependencies and boards want forecasts based on measured flow rather than estimates.

Which roles gain most from Agile Software Delivery Management?

Delivery managers, product leads and engineering heads in software firms gain most from Agile Software Delivery Management. You benefit if you already run Scrum or Kanban teams and now answer for multi-team release dates, technical debt choices and client contract scope.

How does Agile Software Delivery Management differ from a general IT project management course?

Agile Software Delivery Management focuses on product firms running several agile teams on one codebase. A general IT project management course covers stage gates, infrastructure change and mixed methods, while this course concentrates on flow metrics, probabilistic forecasting and agile client contracts.

What do you bring back from Agile Software Delivery Management?

You bring back an agile delivery plan for a real software product from your firm, with a team structure, flow baseline, Monte Carlo release forecast, debt allocation and contract terms. You can present it to your client steering group and update it each release.

Agile Software Delivery Management: The Way Forward

The agile delivery plan first informs which release dates and scope a firm can commit to its client. Delivery managers run cadence from it, product leads order the backlog against it and account leads draft contract terms from it. After the first release, compare forecast with actual throughput and revisit the scaling model and debt allocation where flow fell short.