Project Quality Management Notes

Learning Objectives

- Understand project quality management and why it's important.
- Explain quality planning, assurance, and control processes.
- Identify tools and techniques for quality management.
- Describe how Agile teams ensure product quality step by step.

Content Overview

- Project Quality Management
- Plan Quality Management
- Manage Quality
- Control Quality
- Test Automation
- Test Driven Development

Project Quality Management

- Makes sure project outputs meet needs and what stakeholders expect.
- Includes planning, assurance, and control to check quality.
- Cuts down errors, fixes, and costs with clear rules.
- Encourages ongoing improvement through checks, measurements, and feedback.
- Builds trust with customers and makes the project reliable.
- Boosts success by giving dependable, useful results.

Agile Quality Management

- Builds quality into each short work cycle (sprint/iteration).
- Focuses on quality from the start, not just checking at the end.
- Uses Test-Driven Development (TDD), ongoing code merging, and team reviews to spot issues early.
- Gets regular input from stakeholders to improve outputs.

- Promotes team teamwork and everyone sharing responsibility for quality.
- Improves flexibility, customer happiness, and lowers risks in delivery.

Quality: Traditional vs Agile

Aspect	Traditional	Agile
Focus	Quality as a balance with cost, time, scope (extrinsic quality).	Quality built-in (intrinsic), with constraints like cost, schedule, scope.
Approach	Separate quality checks.	Integrated in every step.

Quality vs Scope

- **Scope**: Defines what the project delivers (features, limits). About the work and needs. Answers: "What will be done?" Part of Work Breakdown Structure (WBS).
- **Quality**: Defines how well deliverables are done, based on standards and expectations. About excellence of output. Answers: "How well will it be done?" Not usually in WBS.

Quality Management Processes

1. Plan Quality Management
2. Manage Quality
3. Control Quality

Plan Quality Management

Traditional View: Set quality needs and rules.

- **Key Inputs**: Project charter, PM plan, stakeholder list, requirements docs, Enterprise Environmental Factors (EEF), Organizational Process Assets (OPA).
- **Tools & Techniques**: Cost-benefit analysis, Cost of Quality (COQ), benchmarking, design of experiments, quality tools (fishbone diagrams, Pareto charts).
- **Outputs**: Quality management plan, quality metrics, updated project docs.

Agile View: Build quality gradually, not as a last check.

- **Key Inputs**: Product vision & roadmap, Definition of Done (DoD), user stories & acceptance criteria.

- **Tools & Techniques**: Agile modeling & ongoing feedback, backlog refinement, Test-Driven Development (TDD).
- **Outputs**: Shared DoD, clear acceptance criteria per user story, team understanding of quality goals.

Cost of Quality Comparison

- Traditional COQ: Prevention (20%), Appraisal (30%), Failure (50%).
- Agile: Higher prevention through automation – shifts to 70% prevention, 20% appraisal, 10% failure.
- Early automation and acceptance criteria cut rework costs.

Manage Quality

Traditional View: Make sure processes hit quality standards (assurance).

- **Key Inputs**: Quality management plan, quality metrics, process improvement plan, performance data.
- **Tools & Techniques**: Quality audits, process analysis, root cause analysis, problem-solving tools.
- **Outputs**: Quality reports, change requests, test and evaluation docs.

Agile View: Add quality to each sprint with teamwork and openness.

- **Key Inputs**: Sprint goals (quality targets), team agreements (standards baseline), Continuous Integration (CI) (process control).
- **Tools & Techniques**: Pair programming, refactoring, Defect Removal Efficiency (DRE), Agile retrospectives, test automation.
- **Outputs**: Shippable product pieces meeting DoD, quality fixes in backlog, shared metrics (build health, defect rate).

Defect Removal Efficiency (DRE)

- Traditional: ~60% (defects found late).
- Agile + CI/CD: ~90% (defects caught early).
- CI + team reviews lower future quality risks.

Control Quality

Traditional View: Check outputs meet needs.

- **Key Inputs**: Deliverables, quality metrics, test results, work performance data, verified changes.
- **Tools & Techniques**: Control charts, inspection, statistical sampling, checklists.
- **Outputs**: Verified deliverables, quality control measurements, change requests.

Agile View: Check quality ongoing during sprint reviews and auto tests.

- **Key Inputs**: Working pieces (deliverables), acceptance criteria (metrics baseline), sprint review feedback (validation input).
- **Tools & Techniques**: Automated regression tests, sprint reviews with stakeholders, Kanban Cumulative Flow, exploratory testing.
- **Outputs**: Accepted user stories/features, real-time defect metrics (defect burn-down per sprint), quick backlog changes.

Test Automation

What/When to Automate

- Regression testing: Run often to check new code doesn't break old (e.g., e-com app).
- Smoke testing: Quick basic checks before deep tests.
- High volume/repetitive tests: Saves time, cuts errors (e.g., load testing).
- Data-driven testing: Same test with many inputs (e.g., 500 name/email combos).
- Cross-browser/device testing: Fast checks on different setups.
- Stable features: For things that don't change much (e.g., login/logout).

What/When Not to Automate

- Exploratory testing: Needs human creativity (e.g., visual bugs).
- Short-lived features: Not worth it for temporary things (e.g., holiday sale).
- Unstable/changing UI: Tests break too often (e.g., promo page).
- Usability testing: Human feedback on feel and ease (e.g., new colors).

- One-time tests: Effort outweighs benefit (e.g., data migration).
- Complex logics: Hard to automate (e.g., image compare, CAPTCHA).

Test Automation Lifecycle

1. **Scope**: Decide what to automate based on complexity, frequency, importance. Work with team.
2. **Choose Tools**: Compare tools by tech support, coding needs, cost, ease, etc. (e.g., Selenium for web, Appium for mobile).
3. **Plan, Design, Strategy**:

Activity	Description	Example
Define Objectives	What to achieve	Speed up regression, improve coverage
Finalize Scope	What to automate/skip	Automate login, skip animations
Select Tools	Fit tech stack	Selenium + TestNG for web
Assess Skills	Team gaps, training	Python training needed
Develop Strategy	High-level plan	Nightly CI runs, page object model
Estimate Timeline/Cost	Time/cost guesses	Setup: 2 weeks, maintenance: 8 hrs/month
Define KPIs	Success measures	Pass rate, coverage

4. **Setup Environment**: App under test, tools, data, browsers/devices, configs, CI/CD link, logs/dashboards.
5. **Test Script & Execution**: Write scripts for test cases; run them (e.g., script for function testing).
6. **Analysis & Reporting**: Collect results, analyze failures, generate reports, visualize metrics (pass/fail rate, flakiness, execution time, coverage, early defects), share insights, improve tests.
    - Example: Online banking app report – 98 tests, 90 passed, 8 failed; 80% coverage; shared on Slack/CI.

Test Automation Pyramid

- Base: Many unit tests (fast, cheap).
- Middle: Some integration/API tests.
- Top: Few UI/end-to-end tests (slow, costly).

Test-Driven Development (TDD)

- Technique that stresses testing first.
- Write requirements as tests before code.
- Write minimal code to pass tests.

Steps

1. Understand feature request.
2. Write a failing test.
3. Write code to pass the test; repeat if needed.
4. Clean code (refactor).
5. Repeat for next requirement.

TDD Cycle (Red-Green-Refactor)

- Red: Test fails (code not working).
- Green: Test passes (works, but not perfect).
- Refactor: Improve code while keeping it working.

Three Rules

1. No code without a failing test first.
2. Write only enough test to fail.
3. Write only enough code to pass the test.

Examples

- Rule 1: Write test for add(2,3)==5 before function; or password validator for length >=6.

- Rule 2: For sum function, start with empty string test ==0, not commas yet.
- Rule 3: Update code minimally for single number, not multiples yet.

Benefits

- Better code: Written to pass tests.
- Quick feedback: Spots breaks fast.
- Good design: Loosely connected, focused code.
- Tests as docs: Show how system works.
- Safe refactoring: Tests catch issues.
- Easier debugging: Isolates problems.

Challenges

- Learning curve: Hard for beginners.
- Slower at start: Tests take time.
- Maintenance: Old tests become burdensome.
- Tough for UI/legacy/non-predictable code.
- Over-testing: Makes changes hard.
- Mindset shift: Think tests first.

Project Resource Management Notes

Learning Objectives

- Explain the importance of Project Resource Management.
- Differentiate between traditional (PMBOK) and Agile approaches.
- Analyze how Agile practices influence resource management.
- Apply resource management concepts to real-world project scenarios.

Content Overview

- Project Resource Management

- Plan Resource Management
- Estimate Activity Resources
- Acquire Resources
- Develop Team
- Manage Team
- Control Resources

Project Resource Management

- Ensures right resources (people, equipment, materials) are available when needed.
- Manages both human (team members) and physical (tangible) resources.
- Optimizes use by balancing efficiency and effectiveness to meet goals.
- Supports team performance through building skills, motivation, and collaboration.
- Strongly linked to scope, schedule, and cost (triple constraints of success).

Agile Resource Management

- Self-organizing teams: Resources (especially people) managed by empowered, cross-functional teams, not a manager.
- Stable, dedicated teams: Teams stay together across iterations for trust, velocity, and predictability, instead of shifting between projects.
- Capacity-based planning: Work planned based on team's sustainable pace and past velocity, not fixed early estimates.
- Servant leadership: Scrum Masters, Product Owners, or Agile leaders enable, coach, and remove blocks, not micromanage.
- Transparency & continuous improvement: Managed via visible boards, WIP limits, and retrospective feedback, not formal reports.

Resource Management Processes

(Diagram showing flow: 1. Plan Resource Management, 2. Estimate Activity Resources, 3. Acquire Resources, 4. Develop Team, 5. Manage Team, 6. Control Resources)

Plan Resource Management

Traditional Perspective

- Importance: Defines roles, responsibilities, reporting lines, staffing approach.
- Key Inputs: Project charter, PM plan, EEF, OPA.
- Tools & Techniques: Org charts, RACI matrix, meetings.
- Outputs: Resource management plan, team charter.

Agile Perspective

- Importance: Focus on self-organizing, cross-functional teams.
- Key Inputs: Product vision, backlog, team norms.
- Tools & Techniques: Team chartering, self-organization, working agreements.
- Outputs: Agile working agreements, Definition of Done, stable team structure.

Comparison

Agile	**Traditional**
Team defines roles collaboratively	PM defines roles and responsibilities
Flexible working agreements	Rigid RACI charts for accountability
Self-organization and collective ownership	Top-down staffing decisions

Organization Structure

(Diagram likely showing types like functional, matrix, projectized organizations)

RACI Matrix

(Diagram of example matrix)

- A – Accountable: Ultimately answerable, holds commercial/legal responsibility, approves projects/tasks, signs off costs, ensures proper completion.
- R – Responsible: Carries out/initiates activity, directly implements, may delegate, includes disciplinary responsibility.

- C – Consulted: Provides input/expertise, consulted for knowledge/experience/info, contributes to decisions/planning.
- I – Informed: Kept updated on progress/results, has right/duty to receive info, not involved in execution/decision-making.

Estimate Activity Resources

Traditional Perspective

- Importance: Identify type & quantity of resources.
- Key Inputs: Scope baseline, activity list, cost estimates.
- Tools & Techniques: Bottom-up estimating, expert judgment, RBS.
- Outputs: Resource requirements, basis of estimates, resource breakdown structure.

Agile Perspective

- Importance: Estimation is iterative, based on team capacity.
- Key Inputs: Backlog items, acceptance criteria.
- Tools & Techniques: Planning poker, velocity tracking, capacity planning.
- Outputs: Sprint capacity, velocity forecast, backlog commitment.

Comparison

Agile	**Traditional**
Capacity estimated per sprint/iteration	Fixed resource hours estimated upfront
Estimates tied to backlog items	Estimates tied to WBS activities
Adaptive, velocity-based planning	Predictive, one-time forecasts

Resource Breakdown Structure (RBS)

(Diagram likely a hierarchical breakdown of resources by category/type)

Acquire Resources

Traditional Perspective

- Importance: Secures team, facilities, equipment at right time.
- Key Inputs: Resource plan, EEF, OPA.
- Tools & Techniques: Negotiation, pre-assignment, decision analysis.
- Outputs: Project team assigned, resource calendars.

Agile Perspective

- Importance: Focus on stable, long-lived, cross-functional teams.
- Key Inputs: Backlog scope, skills needed.
- Tools & Techniques: Self-selection workshops, Agile contracts.
- Outputs: Dedicated Agile team, collaboration tools, team workspace.

Comparison

Agile	Traditional
Teams self-select or form organically	PM negotiates and assigns resources
Dedicated, stable, cross-functional teams	Resources shift across multiple projects
Collaboration-focused Agile contracts	Hiring/contracts focus on scope compliance

Agile Contract

(Diagram likely showing types or examples of Agile contracts)

Develop Team

Traditional Perspective

- Importance: Improves competencies, interaction, performance.
- Key Inputs: HR plan, HR policies.
- Tools & Techniques: Training, team-building, colocation, recognition.
- Outputs: Team performance assessments, skill updates.

Agile Perspective

- Importance: Focuses on continuous learning & collaboration.
- Key Inputs: Team norms, retrospective insights.
- Tools & Techniques: Pair programming, mob programming, servant leadership.
- Outputs: Improved velocity, collaboration, psychological safety.

Comparison

Agile	Traditional
On-the-job learning and retrospectives	Structured training programs planned by PM
Team-building integrated into daily work	Team-building as separate planned events
Peer learning and self-improvement	Manager-driven skill development

Pair Programming

(Diagram on pages 24-25 likely explaining pair programming process)

Types of Pairing

(Diagram on page 26 likely showing different pairing styles like driver-navigator, ping-pong)

Manage Team

Traditional Perspective

- Importance: Monitors performance, resolves conflict, gives feedback.
- Key Inputs: Issue log, work performance reports.
- Tools & Techniques: Leadership styles, conflict management, appraisals.
- Outputs: Change requests, plan updates.

Agile Perspective

- Importance: Self-managing teams, peer accountability, transparent metrics.
- Key Inputs: Stand-ups, sprint reviews.
- Tools & Techniques: Servant leadership, peer feedback, velocity metrics.
- Outputs: Adjusted backlog, improved team dynamics.

Comparison

Agile	Traditional
Team self-manages performance	PM tracks and controls performance
Continuous peer-to-peer feedback	Feedback provided by manager
Conflict resolved collaboratively by team	Conflict resolved by PM authority

Servant Leadership

(Diagram on pages 30-31 likely showing qualities or model of servant leadership)

Control Resources (Note: Text refers to "Control Team", likely a typo for Control Resources)

Traditional Perspective

- Importance: Ensures planned resource usage.
- Key Inputs: Resource assignments, schedule, risk register.
- Tools & Techniques: Performance reviews, audits, variance analysis.
- Outputs: Work performance info, change requests.

Agile Perspective

- Importance: Resources are self-managed by teams.
- Key Inputs: Sprint backlog, velocity trends.
- Tools & Techniques: Kanban boards, swarming, continuous monitoring.
- Outputs: Adjusted WIP limits, backlog reprioritization.

Comparison

Agile	Traditional
Resource usage visualized via Kanban boards	Resource usage tracked via reports/audits
Team swarms to remove bottlenecks	PM reallocates idle staff
Control through transparency & WIP limits	Formal control through audits and reviews

Project Communication Management Notes

Learning Objectives

- Explain why effective communication is key to project success, comparing traditional and Agile methods.
- Use communication tools and techniques in predictive (traditional) and Agile settings.
- Compare structured communication plans vs. flexible Agile feedback loops for engaging stakeholders.

Content Overview

- Project Communication Management
- Plan Communication Management
- Manage Communications
- Monitor Communications

Project Communication Management

- Makes sure the right info gets to the right people at the right time.
- Includes three main processes: Plan Communications Management, Manage Communications, and Monitor Communications.
- Uses tools: In traditional projects - reports, meetings, channels; In Agile - daily stand-ups, info radiators, feedback loops.
- Keeps stakeholders aligned: Cuts misunderstandings, helps decisions, builds trust in teams.

Agile Communication Management

- Focuses on openness and teamwork: Ongoing, team-led talks instead of lots of docs.
- Uses simple tools: Daily stand-ups, Kanban boards, burndown charts, info radiators instead of formal reports.
- Supports quick feedback: Regular reviews, retrospectives, and demos with stakeholders for fast changes.

- Builds trust and ownership: Face-to-face or virtual chats, group decisions, self-organizing teams strengthen bonds.

Communication Management Processes

(Diagram likely showing the flow of the three processes: Plan → Manage → Monitor)

Plan Communication Management

Traditional Perspective

- Aim: Get the right message to the right stakeholder at the right time.
- Key Inputs: Project Charter, Stakeholder Register, PM Plan, EEF, OPA.
- Tools & Techniques: Analyze communication needs, use models, hold meetings.
- Outputs: Communications Management Plan, updated docs.

Agile Perspective

- Aim: Stresses openness, teamwork, and frequent talks.
- Key Inputs: Product Vision, Backlog, Stakeholder Personas, Team Norms.
- Tools & Techniques: Info radiators (Kanban boards, burndown charts), daily stand-ups, tools like Slack or Jira.
- Outputs: Team communication rules, backlog visibility, working agreements.

Comparison

Agile	Traditional
Lightweight team norms & agreements	Formal communication plan upfront
Continuous, flexible communication	Scheduled reporting cycles
Open, transparent dashboards	Hierarchical info flow

Manage Communications

Traditional Perspective

- Aim: Carry out communication activities as planned.
- Key Inputs: Work Performance Reports, Issue Log, Stakeholder Register.

- Tools & Techniques: Communication tech, performance reports, interpersonal skills.
- Outputs: Project communications (reports, presentations, updates), PM plan updates.

Agile Perspective

- Aim: Keeps stakeholders engaged with real-time feedback.
- Key Inputs: Burndown/burnup charts, task boards, sprint reviews.
- Tools & Techniques: Daily stand-ups, sprint reviews, retrospectives, real-time platforms.
- Outputs: Updated backlog, shared dashboards, stakeholder alignment.

Comparison

Agile	Traditional
Real-time dashboards & demos	Periodic status reports
Two-way conversations	One-way communication
Informal, iterative feedback loops	Formal approval cycles

Monitor Communications

Traditional Perspective

- Aim: Check if communications work and meet stakeholder needs.
- Key Inputs: PM Plan, Work Performance Data, Lessons Learned Register.
- Tools & Techniques: Assess stakeholder engagement, meetings, expert judgment.
- Outputs: Work performance info, change requests, project updates.

Agile Perspective

- Aim: Improves communication using feedback and retrospectives.
- Key Inputs: Retrospective notes, stakeholder feedback, velocity metrics.
- Tools & Techniques: Retrospectives, feedback loops, info radiators (CFD, burn charts).

- Outputs: Adjusted backlog, updated working agreements, better communication practices.

Comparison

Agile	Traditional
Communication improved via feedback	Communication monitored via reports
Team & stakeholders evaluate together	PM evaluates effectiveness
Adjustments made iteratively each sprint	Adjustments via formal change requests

Project Stakeholder Management Notes

Learning Objectives

- Explain why identifying, analyzing, and engaging stakeholders matters in traditional and Agile projects.
- Use stakeholder management tools and techniques.
- Compare structured stakeholder plans (traditional) with flexible, team-based engagement (Agile).

Content Overview

- Project Stakeholder Management
- Identify Stakeholders
- Plan Stakeholder Engagements
- Manage Stakeholder Engagements
- Monitor Stakeholder Engagements
- Communication Vs Stakeholder Management

Project Stakeholder Management

- Finds stakeholders: Who the project affects or who can affect it.

- Checks expectations: Looks at their needs, interests, and power levels.
- Engages well: Makes plans to get support and cut resistance.
- Watches relationships: Changes plans as stakeholder needs shift.

Agile Stakeholder Management

- Ongoing teamwork: Stakeholders join in reviews and feedback all through the project.
- Clear visibility: Backlogs, Kanban boards, and demos show progress.
- Flexible engagement: Check priorities often in sprint reviews and planning.
- Group ownership: Decisions made together to build trust and match goals.

Stakeholder Management Processes

(Diagram likely showing flow: Identify → Plan → Manage → Monitor Stakeholders)

Identify Stakeholders

Traditional Perspective

- Focus: Find all people, groups, or orgs affected by the project.
- Key Inputs: Project Charter, Procurement Docs, EEF, OPA.
- Tools & Techniques: Stakeholder analysis, expert judgment, mapping (Power/Interest Grid).
- Outputs: Stakeholder Register.

Agile Perspective

- Focus: Start early and keep collaborating with stakeholders.
- Key Inputs: Product Vision, Backlog, User Stories, Personas.
- Tools & Techniques: Story mapping, customer interviews, workshops, stakeholder canvases.
- Outputs: Updated backlog with stakeholder needs, personas, prioritized stakeholders.

Comparison

Agile	Traditional
Stakeholders engaged continuously	Stakeholders identified at project start
Personas, story mapping for collaboration	Power/interest grids for classification
Product Owner/team maintains living backlog	PM maintains stakeholder register

Stakeholder Mapping

Power-Interest Grid (Diagram on page 10 likely showing quadrants: High Power-High Interest (Manage Closely), High Power-Low Interest (Keep Satisfied), Low Power-High Interest (Keep Informed), Low Power-Low Interest (Monitor))

Stakeholder Analysis Canvas

(Diagram on page 11 likely a template for analyzing stakeholders: Name, Role, Interests, Influence, etc.)

Activity

Identify possible stakeholders for a Vehicle Rental Company's Project for their Booking System based on Power-Interest Grid.

Power-Interest Grid: Example

A Vehicle Rental Company's Project for their Booking System (Diagram on page 13 likely plotting stakeholders like Customers (High Interest-Low Power), CEO (High Power-High Interest), etc.)

Plan Stakeholder Engagement

Traditional Perspective

- Focus: Make strategies to engage stakeholders during the project.
- Key Inputs: PM Plan, Stakeholder Register, EEF, OPA.
- Tools & Techniques: Engagement assessments, expert judgment, decision-making.
- Outputs: Stakeholder Engagement Plan.

Agile Perspective

- Focus: Create engagement strategy together with stakeholders.

- Key Inputs: Backlog priorities, working agreements, team charter.
- Tools & Techniques: Workshops, co-creation sessions, sprint reviews, visual radiators.
- Outputs: Engagement strategies in team norms, backlog transparency.

Comparison

Agile	Traditional
Engagement built into Agile ceremonies	Formal engagement plan documented
Stakeholders participate continuously	PM defines how/when to engage
Adaptive, iterative adjustments	Static strategies

Stakeholder Engagement Assessment Matrix

(Diagram on pages 17-18 likely a table with stakeholders and levels: Unaware, Resistant, Neutral, Supportive, Leading; Current vs. Desired)

Manage Stakeholder Engagement

Traditional Perspective

- Focus: Follow the plan to meet stakeholder expectations.
- Key Inputs: Stakeholder Register, Issue Log, Risk Register.
- Tools & Techniques: Communication methods, negotiation, conflict management.
- Outputs: Issue log updates, change requests, PM plan updates.

Agile Perspective

- Focus: Active teamwork and feedback loops.
- Key Inputs: Backlog, sprint goals, feedback from reviews/demos.
- Tools & Techniques: Sprint reviews, daily interactions, tools (Jira, Slack, Miro).
- Outputs: Updated backlog, alignment with stakeholders, shared ownership.

Comparison

Agile	Traditional
Team & stakeholders collaborate directly	PM manages stakeholder expectations
Engagement via reviews, stand-ups, demos	Engagement through formal updates
Feedback integrated continuously in backlog	Feedback processed via change requests

Monitor Stakeholder Engagement

Traditional Perspective

- Focus: Track relationships and tweak strategies.
- Key Inputs: PM Plan, Stakeholder Engagement Plan, Issue Log, Work Performance Data.
- Tools & Techniques: Engagement assessments, root cause analysis, meetings, expert judgment.
- Outputs: Change requests, work performance info, updates to project docs.

Agile Perspective

- Focus: Keep improving engagement with feedback.
- Key Inputs: Retrospective notes, stakeholder feedback, backlog updates, velocity metrics.
- Tools & Techniques: Retrospectives, sprint reviews, feedback loops, info radiators (CFDs, burnup charts).
- Outputs: Adjusted backlog priorities, updated team agreements, better engagement practices.

Comparison

Agile	Traditional
Stakeholder engagement monitored via direct feedback loops	Stakeholder engagement monitored via reports/surveys
Team + stakeholders evaluate together	PM evaluates stakeholder satisfaction

Agile	Traditional
Adjustments made iteratively during sprints	Adjustments through formal change requests

Feedback Loops

Daily Scrum | Sprint Review | Retrospective | Code Reviews | CICD | TDD

Communication Vs Stakeholder Management

Aspect	Communication Management	Stakeholder Management
Focus	Ensures effective flow of project information	Ensures effective engagement of people influencing/impacted
Processes	Plan, Manage, Monitor communications	Identify, Plan, Manage, Monitor engagement
Key Outputs	Communication plan, reports, performance info	Stakeholder register, engagement plan, status updates
Tools & Techniques	Meetings, reports, dashboards, information radiators	Stakeholder analysis, mapping, engagement assessments
Goal	Deliver right information at right time	Build trust, manage expectations, and gain support

Project Risk Management Notes

Learning Objectives

- Explain the seven risk management processes and their importance in project success.
- Apply traditional and Agile tools & techniques to identify, analyze, and respond to risks.
- Compare how risk management differs between predictive (formal plans, registers) and Agile (continuous, team-driven, adaptive) approaches.

Risk Management - Overview

- Introduction to Risk Management
- Risk Management Processes
- Plan Risk Management
- Identify Risks
- Perform Qualitative Risk Analysis
- Perform Quantitative Risk Analysis
- Plan Risk Responses
- Implement Risk Responses
- Monitor Risks

Introduction to Risks

- Risk: Any uncertain event or condition that, if it occurs, could have a positive or negative impact on a project's objectives.
- Known Risks: Identified and can be analyzed, measured, and planned for. Managed using risk assessments, contingency plans, and mitigation strategies. Allocate contingency reserves. E.g., Supplier delays, Production delays.
- Unknown Risks: Unpredictable, unanticipated, or beyond current knowledge. Managed by building flexibility, resilience, and rapid response systems. Allocate management reserves. E.g., Pandemic situations.
- Inherent Risks: Level of risk before any controls, mitigation, or corrective actions are applied. Represents natural exposure to uncertainty. E.g., A web application handling user data inherently risks cyberattacks or data breaches.
- Residual Risks: Remaining risk after mitigation measures, controls, and safeguards have been implemented. Represents risks that cannot be eliminated but must be accepted, monitored, or managed. E.g., After implementing firewalls, encryption, and MFA, small chance of zero-day vulnerabilities.
- Risk Attitude: Approach to risk-taking influencing decision-making.
    - Risk-Averse: Avoids risk, prioritizes safety, stability, certainty. E.g., Investor choosing government bonds over stocks.

- Risk-Neutral: Decisions based on expected outcomes, uninfluenced by uncertainty. E.g., Business choosing project with highest expected profit.
- Risk-Seeking: Takes high risk for greater rewards. E.g., Startup entrepreneur investing in unproven idea.

Risks Vs Issues

- Probability of a Risk is less than 1 while probability of an issue is equal to 1.
- Issue = Realized Risk.

What is Risk Management

- Structured process encompassing planning, identification, analysis, response planning, and monitoring and control of risks to ensure project objectives are achieved.
- Includes the processes listed in overview.

Plan Risk Management

Traditional Perspective

- Focus: Defines how risk management will be carried out.
- Inputs: Charter, PM plan, EEF, OPA.
- T&T: Expert judgment, data analysis, meetings.
- Outputs: Risk management plan.

Agile Perspective

- Focus: Risk strategy emerges from team discussions.
- Inputs: Product backlog, team charter, working agreements.
- T&T: Agile spikes, risk-adjusted backlog, retrospectives.
- Outputs: Team norms for handling uncertainty, embedded in backlog and DoD.

Comparison

Agile	Traditional
Risks integrated into backlog & reviews	Separate formal risk plan

Agile	Traditional
Team-driven discussions	PM-driven planning
Emergent, evolving categories	Defined categories upfront

Agile Spikes

- What is an Agile Spike? A time-boxed user story/task for research, exploration, or prototyping. Purpose: reduce uncertainty or risk before implementation.
- Types: Technical Spike (explore how to build, e.g., test a new API). Functional Spike (explore what to build, e.g., validate UI design).
- Spikes in the Sprint: Treated as a backlog item, added to sprint backlog. Time-boxed (e.g., 1–2 days). Output: knowledge, decision, or prototype (not production code).
- Helps team: Refine estimates, clarify requirements, reduce technical and business risks.

Identify Risks

Traditional Perspective

- Focus: Systematically documents risks.
- Inputs: Scope, schedule, cost plans, assumptions.
- T&T: Root Cause Analysis (RCA), Brainstorming, SWOT, interviews.
- Outputs: Risk register.

Agile Perspective

- Focus: Continuous identification during ceremonies.
- Inputs: Backlog, user stories, sprint goals.
- T&T: Daily stand-ups, retrospectives, risk burndown chart.
- Outputs: Visible risks on Kanban board, updated backlog.

Comparison

Agile	Traditional
Continuous in stand-ups	Risk workshops, documents

Agile	Traditional
Risks on boards, burndowns	Risk register maintained
Lightweight, team-driven	Formal analysis

Risk Burndown Chart

- Risk exposure = Probability × Impact.
- (Diagram likely showing chart)

RCA: 5-Whys Techniques

- Example: Why is the project behind schedule? Developers missed milestones. Why? Repeated code revisions and bug fixes. Why? Requirements unclear and changed frequently. Why? Stakeholders provided conflicting inputs. Why? No formal process for requirement gathering and approval.
- Root Cause: Lack of a structured requirements management process.

RCA: Fishbone (Ishikawa) Diagram

- (Diagram)

Risk Breakdown Structure

- Hierarchical chart that organizes and categorizes risks, aiding in their identification, assessment, and communication.
- (Diagram likely)

Risk Register

- Document that lists, tracks, and manages identified project risks.
- (Diagram likely)

Perform Qualitative Risk Analysis

Traditional Perspective

- Focus: Prioritizes risks by probability/impact.
- Inputs: Risk register, assumptions, stakeholder tolerance.
- T&T: Probability-impact matrix, expert judgment.
- Outputs: Updated risk register with rankings.

Agile Perspective

- Focus: Team assesses risks collaboratively.
- Inputs: Backlog risks, sprint goals.
- T&T: Planning poker for risk scoring, relative sizing, risk burndown.
- Outputs: Prioritized backlog reflecting risk exposure.

Comparison

Agile	Traditional
Risk scoring via team estimation	Probability-impact matrices
Lightweight, iterative adjustments	Document-heavy analysis
Whole team collaborates in prioritization	PM/stakeholders rank risks

Probability and Impact Matrix

- Helps calculate Inherent Risk Rating of a risk.
- Inherent Risk: Measure of a risk, calculated by its probability and impact.
- Probability: Likelihood that a risk occurs. Scale: High, Medium, Low.
- Impact: Damage a risk could cause. Scale: High, Medium, Low.
- (Matrix diagram)

Perform Quantitative Risk Analysis

Traditional Perspective

- Focus: Uses numerical techniques to assess impact.
- Inputs: Risk register, cost/schedule data.
- T&T: Monte Carlo simulation, decision trees, sensitivity analysis.
- Outputs: Quantified risk exposure, contingency reserves.

Agile Perspective

- Focus: Simplified numerical analysis in backlog management.
- Inputs: Velocity, cycle time, backlog estimates.

- T&T: Velocity forecasting, simulation with burn-up/burndown charts, risk-adjusted velocity.
- Outputs: Forecast ranges, buffers integrated into sprint planning.

Comparison

Agile	Traditional
Simple forecasting with velocity	Heavy simulations
Charts and empirical metrics	Specialized tools
Adaptive buffers in sprint planning	Formal reserve planning

Sensitivity Analysis – Tornado Chart

- (Diagram)

Decision Tree Analysis

- Scenario example: Online delivery platform preparing for festive season. Options: Build new infrastructure (cost 12M, strong demand 20M, weak 9M) or Upgrade (cost 5M, strong 12M, weak 6M). Strong demand prob 60%.
- Using EMV: Build EMV = 0.6*8M* + *0.4*(-3M) = 3.6M. Upgrade = 0.6*7M* + *0.4*1M = 4.6M. Choose upgrade.
- (Diagram with decision nodes, chance nodes, ends)

Plan Risk Responses

Traditional Perspective

- Focus: Defines actions to address risks.
- Inputs: Risk register, risk management plan.
- T&T: Strategies (avoid, transfer, mitigate, accept).
- Outputs: Updated risk register, risk response plan.

Agile Perspective

- Focus: Responses embedded in backlog and ceremonies.
- Inputs: Sprint backlog, DoD, retrospective outcomes.
- T&T: Adjusting backlog items, adding spikes, definition of done, test automation.

- Outputs: Risk-mitigating backlog items, shared action items.

Comparison

Agile	Traditional
Responses built into backlog items	Risk response plan
Team-driven collaborative strategies	PM-driven strategies
Continuous adaptation in sprints	Separate risk register updates

Risk Response Strategies

For Negative Risks (Threats):

- Avoidance: Eliminate entirely by changing project plan (e.g., alternative supplier).
- Mitigation: Reduce probability or impact (e.g., using backup suppliers).
- Transfer: Shift to third party (e.g., insurance, outsourcing).
- Acceptance: No proactive action but plan contingency reserves.

For Positive Risks (Opportunities):

- Exploit: Ensure opportunity occurs (e.g., assigning top talent).
- Enhance: Increase probability or impact (e.g., automating repetitive jobs).
- Share: Partner with another entity (e.g., with a foreign party).
- Accept: Recognize but not actively pursue.

Implement Risk Responses

Traditional Perspective

- Focus: Executes risk response plans.
- Inputs: Risk response plan, PM plan.
- T&T: Project management software, change requests.
- Outputs: Implemented risk actions, change logs.

Agile Perspective

- Focus: Teams act immediately during sprints.

- Inputs: Sprint backlog, daily updates.
- T&T: Stand-ups for tracking risk actions, visual boards, swarming.
- Outputs: Updated backlog with mitigations, visible progress.

Comparison

Agile	Traditional
Immediate adjustment within sprints	Follows pre-defined response
Adapt backlog directly	Separate change requests
Actions tracked via boards & stand-ups	Documented actions

Monitor Risks

Traditional Perspective

- Focus: Tracks existing risks, identifies new ones, evaluates effectiveness.
- Inputs: Risk register, work performance data.
- T&T: Audits, variance analysis, status meetings.
- Outputs: Updated risk register, lessons learned.

Agile Perspective

- Focus: Continuous monitoring during ceremonies.
- Inputs: Sprint backlog, burn-down data, retrospective feedback.
- T&T: Risk burndown charts, daily check-ins, retrospectives.
- Outputs: Updated backlog, adapted working agreements, shared learning.

Comparison

Agile	Traditional
Continuous monitoring in sprints	Periodic monitoring
Lightweight ceremonies	Formal reviews & audits
Visible charts, backlog adjustments	Risk register updates

Project Procurement Management Notes

Learning Objectives

- Explain why effective procurement management matters in software projects.
- Describe and compare the 3 procurement management processes in traditional and Agile contexts.
- Apply Agile procurement ideas like vendor collaboration, adaptive contracts, and value-based delivery to real-world projects.

Content Overview

- Project Procurement Management
- Plan Procurement Management
- Conduct Procurements
- Control Procurements

Project Procurement Management

- Handles getting goods and services from outside to complete the project.
- Includes three main processes: planning, conducting, and controlling procurements.
- Makes sure everything follows legal and financial rules using contracts, agreements, and vendor checks.
- In Agile, focuses on teamwork: Uses flexible, value-focused contracts and ongoing vendor feedback instead of strict docs.

Agile Procurement Management

- Puts collaboration above contracts: Builds trust and common goals with vendors rather than fixed rules.
- Uses adaptive contracts: Agreements can change, letting scope and outputs evolve over sprints.
- Brings vendors into the team: Suppliers join sprint reviews and planning for openness and matching goals.
- Stresses ongoing feedback: Reviews vendor work step by step to boost value and quick responses.

Procurement Management Processes

(Diagram likely showing the flow: Plan → Conduct → Control Procurements)

Plan Procurement Management

Traditional Perspective

- Focus: Sets how to get external goods/services.
- Key Inputs: Project charter, scope baseline, cost estimates, risk register, EEF, OPA.
- Tools & Techniques: Make-or-buy analysis, market research, expert judgment, meetings.
- Outputs: Procurement management plan, procurement strategy, bid documents, source selection criteria, make-or-buy decisions.
- Example: Outsourcing cloud hosting after cost-benefit check.

Agile Perspective

- Focus: Teamwork with flexible partners instead of strict contracts.
- Key Inputs: Product vision, prioritized backlog, vendor capabilities, team needs.
- Tools & Techniques: Collaborative vendor workshops, team chartering, iterative decision-making, supplier demos.
- Outputs: Working agreements, lightweight procurement plan, Definition of Done (DoD).
- Example: Hiring a DevOps vendor with sprint-based reviews and contract tweaks.

Comparison

Traditional	Agile
Fixed, detailed procurement plans	Flexible and iterative vendor engagement
Emphasis on contract compliance	Emphasis on collaboration and shared value
Decisions made at project start	Decisions revisited as backlog evolves

Conduct Procurements

Traditional Perspective

- Focus: Get bids, review proposals, award contracts.
- Key Inputs: Procurement documents, seller proposals, resource calendars, cost baseline.
- Tools & Techniques: Bidder conferences, proposal evaluation, negotiation, advertising.
- Outputs: Selected sellers, signed agreements, resource updates, change requests.
- Example: Picking a software vendor via RFP review and formal contract.

Agile Perspective

- Focus: Use teamwork and prototypes instead of long proposal rounds.
- Key Inputs: Sprint goals, backlog items, vendor capabilities, product vision.
- Tools & Techniques: Joint workshops, trial sprints, vendor demos, time-boxed negotiations.
- Outputs: Shared backlog, iterative contract agreements, feedback loops.
- Example: Doing a short pilot sprint to test a UI/UX design partner before full commitment.

Comparison

Traditional	Agile
Vendor selection based on fixed proposals	Selection through working prototypes
Long negotiation cycles	Rapid, time-boxed collaboration
Contract awarded for full scope	Incremental, value-based contracting

Control Procurements

Traditional Perspective

- Focus: Checks vendor performance and contract rules.
- Key Inputs: Agreements, procurement docs, performance reports, change requests.

- Tools & Techniques: Contract control systems, audits, inspections, performance reviews.
- Outputs: Closed procurements, work performance info, updated documents.
- Example: Looking at monthly vendor reports and processing payments.

Agile Perspective

- Focus: Blends vendor checks into sprint work and reviews.
- Key Inputs: Agile contracts, sprint deliverables, performance metrics, feedback.
- Tools & Techniques: Sprint reviews, retrospectives, backlog refinement, collaborative issue resolution.
- Outputs: Continuous vendor performance feedback, adaptive contracts, lessons learned, improved team alignment.
- Example: Evaluating vendors regularly via sprint demos, not formal quarterly audits.

Comparison

Traditional	Agile
Performance reviews at fixed intervals	Continuous feedback via sprint reviews
Contract changes through formal requests	Backlog reprioritization replaces change control
Focus on compliance	Focus on relationship and value delivery

Procurement Management Summary

Process	Traditional Focus	Agile Focus
Plan Procurement	Formal planning and documentation	Adaptive collaboration and vendor flexibility
Conduct Procurements	Vendor selection and contracting	Iterative engagement and shared delivery
Control Procurements	Monitoring performance and compliance	Continuous feedback and adaptive relationships