

# Minimal Practical Agile Software Engineering Process

Emmanuel BRUNO  
Université de Toulon, LIS UMR CNRS 7020  
emmanuel.bruno@univ-tln.fr

2026-10-03

## Table des matières

|                                                              |   |
|--------------------------------------------------------------|---|
| 1 Minimal Practical Agile Software Engineering Process ..... | 1 |
| 1.1 1. Project Initiation .....                              | 1 |
| 1.1.a User Story Template .....                              | 2 |
| 1.2 3. Agile Development Workflow ....                       | 2 |
| 1.2.a Option A: Scrum Sprints .....                          | 2 |
| 1.2.b Option B: Kanban Workload ...                          | 2 |
| 1.3 4. Quality Assurance .....                               | 2 |
| 1.3.a Acceptance Criteria (Gherkin Syntax) .....             | 2 |
| 1.4 5. Release & Operations .....                            | 2 |
| 1.5 6. Roles & Responsibilities .....                        | 3 |
| 1.6 7. Risk Management (Minimal) .....                       | 3 |
| 1.7 8. Feedback Loops & Continuous Improvement .....         | 3 |
| 1.8 9. Stakeholder Engagement .....                          | 3 |
| 1.9 10. Minimal Documentation Guidance .....                 | 3 |
| 1.10 11. Security & Compliance Reminder .....                | 3 |
| 1.11 12. Tooling Recommendations ....                        | 3 |
| 1.12 7. Core Principles .....                                | 3 |
| 1.13 8. Minimal Artifacts Checklist ....                     | 3 |

## 1 Minimal Practical Agile Software Engineering Process

This guideline distills the essentials of a robust, real-world agile software engineering process. It is designed for teams seeking a clear, actionable, and adaptable workflow from project initiation to release and continuous improvement.

### 1.1 1. Project Initiation

- **Define Vision & Scope:** Establish project goals, stakeholders, and high-level requirements.
- **Form the Team:** Assign roles (Product Owner, Scrum Master, Developers, QA, etc.).
- **Kickoff Meeting:** Align on objectives, constraints, and communication channels.
- **Deliverables:** Project Charter (template)
  - Team Structure (guide)

**Elicit Requirements:** Use interviews, workshops, and document analysis. **Document Requirements:** Capture functional, non-functional, and user stories. **Prioritize & Plan:** Build and groom the product backlog. Define MVP. **Deliverables:** - System Requirements (template) - User Stories (template) - Technical Design (template)

### 1.1.a User Story Template

As a <type of user>, I want <some goal> so that <some reason>.

#### Example:

As a registered user, I want to reset my password so that I can regain access if I forget it.

## 1.2 3. Agile Development Workflow

You can choose between two main agile workload management styles:

### 1.2.a Option A: Scrum Sprints

- **Iterative Sprints:** Plan, execute, and review work in short cycles (1-4 weeks).
- **Daily Standups:** Synchronize team progress and address blockers.
- **Sprint Review & Retrospective:** Inspect increment, adapt process.
- **Deliverables:**
  - Sprint Backlog (guide)
  - Development Guidelines (template)

### 1.2.b Option B: Kanban Workload

- **Continuous Flow:** Work items are pulled as capacity allows, no fixed-length sprints.
- **Visualize Workflow:** Use a Kanban board to track status (To Do, In Progress, Done, etc.).
- **Limit Work in Progress (WIP):** Prevent overload and improve flow.
- **Regular Check-ins:** Daily standups and periodic reviews for improvement.
- **Metrics:** Track lead time, cycle time, and throughput.
- **Deliverables:**
  - Kanban Board (guide)
  - WIP Policies (template)
  - Development Guidelines (template)

### 1.2.b.a Definition of Done (DoD) Checklist

- ☐ Code implemented and reviewed
- ☐ Unit and integration tests pass
- ☐ Acceptance criteria met
- ☐ Documentation updated
- ☐ Deployed to staging/demo environment

## 1.3 4. Quality Assurance

- **Testing Pyramid:** Emphasize unit, integration, and UI tests.
- **Automated Testing:** Integrate tests into CI pipeline.
- **Code Reviews:** Peer review for quality and knowledge sharing.
- **Metrics:** Track code coverage, defect rates, and build health.
- **Deliverables:**
  - Quality Assurance Plan (template)
  - Test Strategy (template)

### 1.3.a Acceptance Criteria (Gherkin Syntax)

```
Scenario: Resetting a forgotten password
  Given I am on the login page
  And I have forgotten my password
  When I click on "Forgot Password"
  And I enter my registered email address
  Then I receive a password reset link by email
  And I can set a new password
```

## 1.4 5. Release & Operations

- **Release Planning:** Define release criteria and schedule.
- **Deployment Automation:** Use CI/CD for reliable releases.
- **Monitoring & Support:** Track performance, errors, and user feedback.
- **Continuous Improvement:** Hold retrospectives and refine processes.

- **Deliverables:**
  - Release Plan (template)
  - Deployment Guide (template)

## 1.5 6. Roles & Responsibilities

- **Product Owner:** Owns backlog, prioritizes features, interfaces with stakeholders.
- **Scrum Master:** Facilitates agile ceremonies, removes impediments.
- **Development Team:** Delivers increments, ensures quality.
- **QA/Tester:** Designs and executes tests, ensures product quality.

## 1.6 7. Risk Management (Minimal)

- Identify top 3 project risks at initiation and review at each iteration.
- Track risks in a simple log: | Risk | Impact | Likelihood | Mitigation | Status | |---|---|  
-----|-----|---| | Example: Key staff unavailable | High | Medium | Cross-train team | Open |

## 1.7 8. Feedback Loops & Continuous Improvement

- Hold a retrospective at the end of each iteration or monthly for Kanban.
- Capture 1-2 improvement actions and assign owners.
- Incorporate feedback into the next cycle.

## 1.8 9. Stakeholder Engagement

- Schedule regular demos or reviews with stakeholders (at least once per iteration).
- Collect feedback and update backlog accordingly.

## 1.9 10. Minimal Documentation Guidance

- At each phase, maintain only essential documentation:
  - **Initiation:** Project Charter, Team Structure

- **Requirements:** User Stories, Acceptance Criteria, Minimal Requirements Spec
- **Design:** Simple architecture diagram, API/interface sketch
- **Development:** Inline code comments, README, API doc (if needed)
- **Release:** Release notes, deployment instructions

## 1.10 11. Security & Compliance Reminder

- Consider security, privacy, and compliance requirements from the start.
- Add a checklist item for security review before release.

## 1.11 12. Tooling Recommendations

- **Version Control:** Git (GitHub, GitLab, etc.)
- **Work Tracking:** Kanban board (Trello, Jira, GitHub Projects)
- **CI/CD:** GitHub Actions, GitLab CI, or similar
- **Documentation:** Markdown, Quarto, or Confluence
- **Communication:** Slack, Teams, or email

## 1.12 7. Core Principles

- **Agile Values:** Individuals and interactions, working software, customer collaboration, responding to change.
- **Engineering Excellence:** TDD, CI/CD, code quality, documentation, security.
- **Transparency & Adaptation:** Visualize work, measure progress, adapt based on feedback.

## 1.13 8. Minimal Artifacts Checklist

- ☐ Project Charter
- ☐ Team Structure
- ☐ System Requirements Specification
- ☐ User Stories
- ☐ Technical Architecture

- ☐ Sprint Backlog
  - ☐ Development Guidelines
  - ☐ Quality Assurance Plan
  - ☐ Test Strategy
  - ☐ Release Plan
  - ☐ Deployment Guide
- 

See [/02\\_SE\\_101\\_L\\_sections/](#) for detailed templates and guides for each phase. For a full process, refer to Framework Overview and Course Introduction.

**Tip:** Adapt this process to your team's context. Start simple, iterate, and continuously improve.