AGILE INTERMEDIATE 2 DAYS

Advanced Certified Scrum Developer (A-CSD)

The Advanced Certified Scrum Developer (A-CSD) is the second level of the Developer track. It is designed for developers with at least one year of experience on Scrum teams who want to deepen their technical Agile practices, improve workflow visibility, evolve their Definition of Done, and build robust CI/CD pipelines. The course emphasises hands-on application of advanced engineering techniques.

Why A-CSD?

Recognised credential

Advanced Certified Scrum Developer (A-CSD) credential from Scrum Alliance, valid for two years.

Practical skills

Techniques to evolve team quality standards through an improved Definition of Done.

Career progression

Ability to implement and sustain advanced Agile engineering practices on real projects.

Career progression

Foundation to progress to the Certified Scrum Professional - Developer (CSP-D).

Practical skills

Skills to build, maintain, and improve CI/CD pipelines.

What you'll be able to do

- ✓ Visualise and optimise development workflows using Kanban and related tools
- ✓ Evolve and strengthen the team's Definition of Done
- ✓ Evaluate and systematically improve code and product quality
- ✓ Apply advanced refactoring techniques to improve existing codebases
- ✓ Implement and apply Test-Driven Development in a sustained way
- ✓ Build and manage Continuous Integration and Continuous Delivery pipelines
- ✓ Apply behaviour-driven development (BDD) practices
- ✓ Identify and reduce bottlenecks in the development process

Who this is for

- Developers
- Software Engineers
- QA Engineers
- DevOps Engineers
- Technical Leads
- Scrum team members

Curriculum

01 Advanced Agile Engineering Mindset

- Technical excellence as a precondition for business agility
- The role of engineering practices in enabling fast, sustainable delivery
- Technical debt: types, causes, measurement, and repayment strategies
- Team ownership of quality and the culture of craftsmanship
- eXtreme Programming (XP) values and practices

02 Visualising Workflows

- Kanban principles and practices for development teams
- Building and using Kanban boards alongside Scrum
- Cumulative Flow Diagrams: reading and acting on them
- Cycle time and lead time analysis
- Identifying and removing flow bottlenecks
- Work In Progress (WIP) limits and their impact on delivery

03 Evolving the Definition of Done

- Auditing the current Definition of Done
- Identifying quality gaps and adding missing criteria
- Automation as a route to a stronger Definition of Done
- Connecting the Definition of Done to the Increment and release readiness
- Aligning the Definition of Done across multiple teams

04 Evaluating and Improving Quality

- Code quality metrics: cyclomatic complexity, code coverage, duplication
- Static analysis tools and how to integrate them into pipelines
- Code review practices: what to look for, what to automate
- Non-functional requirements: performance, security, and scalability
- Quality at speed: balancing quality practices with Sprint velocity

05 Advanced Refactoring

- Refactoring large codebases incrementally
- Strangler fig pattern for legacy system modernisation
- Safe refactoring sequences: tests first, then change
- Working with legacy code (no tests, poor structure)
- Micro-refactoring as a daily habit

06 Applied Test-Driven Development

- Behaviour-Driven Development (BDD): Given-When-Then format
- Acceptance Test-Driven Development (ATDD)
- Test pyramid: balancing unit, integration, and end-to-end tests
- Mocking, stubbing, and test doubles
- Maintaining test suites: avoiding test rot and flakiness

07 Continuous Integration and Delivery in Practice

- Designing a CI/CD pipeline for a Scrum team
- Build servers: Jenkins, GitHub Actions, GitLab CI
- Automated deployment strategies: blue-green, canary, rolling
- Feature flags and progressive delivery
- Monitoring and observability as part of the delivery cycle