SCRUM ALLIANCE® ADVANCED CERTIFIED SCRUM DEVELOPER SM (A-CSD SM)
Learning Objectives Examples

August 2021

1 - Lean, Agile & Scrum

1.1 Example: Scrum Board, Kanban Board.

1.2 Example: Theory of Constraints, Limiting Work-in-Progress, Flow, Queue theory, Value Stream Mapping

1.3 Example: TIMWOOD (Waste) - Transport, Inventory, Movement, Waiting, Overproduction, Overengineering, Defect, Duplicate work could be addressed by Collective Ownership of Design and Code and Product.

1.4 Example: Conduct a DoD workshop. Reasons why and how the DoD should evolve: New insights, rising quality demands, changes to organizational policies. The DoD could evolve by addition: new attributes, tests or tasks are added. It could evolve by reduction: ultimately, “done” means “done”, as in “no remaining work required to provide customer value”.

1.5 Example: SoS, CoPs, Shared PBL, Shared DoD, Scaling/Descaling (Multiple Teams etc.)

1.7 Example: Cost-Value-Analysis, Value-Risk-Analysis, Page Object Pattern, Test Maintainability, How much money we would like to spend on each level of Mike Cohn’s Test Pyramid.

2 - Collaboration & Team Dynamics

2.1 Example: Cooperation, collaboration and co-creation (Collective Ownership/Stewardship), Self-managing vs. self-directing teams, IBM surgical team, jazz bands (pickup groups)

2.2 Example: Retrospective with story cubes,: (3 Levels of listening, Active Listening)

2.3 Example: Using a feedback structure, Non-Violent Communication, feedback vs. criticism, clean language, Crucial Conversations, focused conversation.

2.4 Example: Pair/mob/ensemble programming.

3 - Architecture & Design

3.1 Examples: emergent architecture: reversibility, decide at the last responsible moment, cross-functional architecture, share architectural decision knowledge - [Building Evolutionary Architecture]

3.2 Example: SOLID design principles, Pragprog, Beck’s Four Rules of Simple Design
Books: [*Pragmatic Programmer* - Hunt & Thomas], shared language or metaphor.

3.3 Example: BDD/ATDD, Event storming, Load-Testing etc.

3.4 Example: Cyclomatic complexity, test coverage, unit length, number of warnings, WTFs per minute, static analysis. Books to go deeper: McConnell’s *Code Complete, Clean Code* - Martin, *Design Patterns* - GoF

4 - Refactoring

4.1 Example: Strangulation, Anti-corruption layer.

4.2 Example: Repetitions: DRY, overly long functions: extract blocks, poorly chosen variable names: rename

4.3 Example: Why is refactoring crucial? What is the importance of tests? What are baby-steps.

4.4 Example: pressure, deadlines, no collective code and product ownership/stewardship, lack of testing, lack of concern, offshoring, lack of slack/buffer time.

5 - Test Driven Development (TDD)

5.1 Example: Baby steps, Build Quality In, Defer Commitment, Fast feedback loops.

5.2 Example: A kata or a small application would be sufficient here. Something like the three rules by Robert Martin should be practiced here.

5.3 Example: Apply Red-Green-Refactor-Cycle, Arrange-Act-Assert, Avoid Conditionals and Loops, Small tests.
Arrange: take care of the preparations, Act: perform the action to be verified, and Assert: compare the actual to the expected results. (From TDD Beck?)
Given, When, Then; Preconditions, actions, verification (or Post-conditions).

5.4 Examples: Test smells, Quality criteria: fast, well-factored, single purposed, self- explanatory, independent. Three distinct phases: precondition, execution, validation, how to design a test suite for maintainability, naming principles. -> see [xUnit Testing Patterns]

5.5 Example: Testing pyramid (M. Cohn), Four test-quadrants (Brian Marick)

5.6 Examples: ubiquitous language, fast feedback cycles, business readable, automatable acceptance tests, living documentation.

5.7 Examples: Implement the ATDD 4D steps and map the steps to the Scrum events and activities, create automated acceptance tests in Gherkin or Fitnesse or any framework of your choice.

5.8 Examples: Use mocks, stubs etc., apply Legacy Characterization Testing [Michael Feathers]

5.9 Non-functional requirements load behavior and load testing, Fitness Functions. Quality: the sum of external and internal system attributes. Internal: Not visible or measurable by the customer, external: visible and obviously relevant. Technical Excellence = Inner Quality * Improvement Capability.

LEARNING OBJECTIVES

6 - Integrating Continuously

6.1 Examples: Feedback time, validation, integration of the work of multiple teams, deployment, branching strategy
6 + 1 disciplines of CI (Book: *Continuous Integration* by Paul Duvall)

CI rules by [Martin Fowler].

6.3 Example: It is totally fine to work with a framework here. Include optimization of setup, compile, build time.

6.4 Example: Integrate a product with two or more product modules, include unit testing, add acceptance and regression testing. Taking responsibility, that the complete product works for the user.

7 - Learning by Delivering Continuously

7.1 Example: “continuous delivery” means that you are ready and able to deploy any version to any supported platform at any time, so that you can create customer value sooner, get better feedback sooner.

7.2 Examples: Feature flags, automated deployment pipelines, infrastructure as code and dynamic infrastructure, convention over configuration, canary releases.

7.3 Examples: Measure increase of app downloads, revenue changes, performance metrics, uptime improvements.

PROGRAM TEAM

Certified Scrum Developer Team (2021)

- Axel Wilhelm Berle
- Pascal Gugenberger
- Björn Jensen
- Falk Kühnel
- Paul Moore
- Rob Myers
- Andreas Schliep
- Rickard Jones