SCRUM ALLIANCE® CERTIFIED SCRUM DEVELOPER (CSD)
Learning Objectives Examples

April 2021

1 - Lean, Agile & Scrum

1.1 This should include creating a sprint goal, selecting product backlog items (PBIs) with a value orientation in mind and decomposing PBIs into smaller work items.

1.2 This emphasizes the collaboration between the members of the Scrum team, especially product owner and developers, for understanding the items and turning them into a valuable stepping stone toward the product goal.

1.3 Methods to organize the daily Scrum could be: original three questions, walk the wall, collaborative daily Scrum.

1.4 Description, order, size, domain-specific attributes.

1.5 This could include asking for clarification, adding acceptance criteria, sizing or estimating items, breaking them down into smaller items.

1.6 For software, a definition of done could state that each increment fulfills the acceptance criteria, contains no blocking defects, is integrated into the system, is properly tracked in a version control system, and is documented according to the necessary documentation guidelines. There could be other elements, too. The point of this LO is to raise the awareness of done, especially “done for whom?”.

2 - Collaboration & Team Dynamics

2.1 This could include stable membership, reaching a shared goal by working together, self-management.

2.2 Example definition of team: small number of people with complementary skills, committed to a common purpose, performance goal and approach, for which they are mutually accountable.Example: T-shaped skill profile.

2.3 Pair programming, pair designing, pair working in general. Creating something – code or non-technical – in a Scrum simulation.

2.4 Shorter feedback loops, less work-in-progress, less mis-interpretation of needs, direct feedback of the working solution, actually trying the built product

2.5 During sprint review while obtaining feedback regarding the latest increment, during sprint review while identifying what kind of changes to the product backlog are a helpful response to changing circumstances, during product backlog refinement, during a story mapping workshop, observing users using the product, interview users how the currently solve their problem, “friendly users” actively using the new product and giving feedback, invite to the sprint review.

3 - Architecture & Design

3.1 Foster understanding of the work, improve shared ownership and responsibility, improve estimability and reliability, reduce defects, avoid accumulation of technical debt

3.2 This could include test first, quick design sessions, using spikes to understand a new technology, constant refactoring, metaphor, conversational modelling, CRC cards, pair programming.

3.3 Reversibility, KISS; DRY; YAGNI; decide at the last responsible moment; responsive, resilient, elastic, message-driven (Reactive Manifesto); S.O.L.I.D.

4 - Refactoring

4.1 Refactoring is the practice of improving a system without changing it's observable behavior.

4.2 Refactoring fosters automated testing, improves readability and maintainability; it can support improving system performance, counter technical debt, and improve extensibility.

5 - Test Driven Development (TDD)

5.1 Test first focuses on the behavior of the system. The system design emerges dynamically through the adjustment to additional tests. Benefits include simplicity, avoidance of bloating, defect reduction, built-in regression checks with automated tests.

5.2 Differences include frequency, inclusion in the development process, being tied to specification versus being tied to code, amount of automation, responsibility, timing.

5.3 Point out that it is important to have the simplest possible solution that passes the test in the first place to ensure the test is working, and refactor it to a sensible design that can be verified immediately, refer to the Broken Window metaphor, foster collective stewardships.

5.4 Testers are Scrum team members, aka developers, include testing in the sizing discussion, collaborate on PBI refinement and acceptance criteria definition, pair up doing TDD to embed good tests from the beginning, include considerations of load, explorative, and other testing approaches, contribute to definition of done.

LEARNING OBJECTIVES

6 - Integrating Continuously

6.1 CI is the practice of reducing the feedback time between code creation and integration. It can be supported using technical means. CI changes habits and behaviors, creates awareness about shared responsibility, is the base for frequent deliveries, reduces defects, reduces risk from late discovery of integration issues.

6.2 Example 1: The increment is necessarily integrated. Using CI is highly recommendable for getting to at least one increment per sprint or more.

Example 2: The daily Scrum can provide better transparency about the progress if there is a better measure for progress. CI helps to get this transparency about what is working and what isn't. Without CI, the Scrum team might probably not know the actual status of the system for days.

Example 3: In the sprint review, only done PBIs are inspected. CI improves the probability of having done PBIs at the end of the sprint.

6.3 Automation tools reduce manual repetition, single repository acts as document of record, configuration as code for repeatable error-free process and support continuous delivery

PROGRAM TEAM

Certified Scrum Developer Team (2021)

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