SCRUM ALLIANCE® CERTIFIED SCRUM PROFESSIONAL® - DEVELOPER (CSP®-D)
Examples

November 2021

EXAMPLES

1 - Enabling a Culture of Technical Excellence

1.1 Reduce cost of change, reduce defect count, enable faster feedback cycles.[1]

1.2 Coaching is a support process for personal and professional growth. Technical coaching combines technical expertise, coaching, mentoring and teaching to help an individual or group to adopt and adapt technical practices in pursuit of excellence.[2]

1.3 List the elements of a coaching agreement, present a completed agreement during coaching, craft an agreement with other students.[3]

1.4 Pick a technical topic and coach, mentor, train or facilitate the individuals or teams to improve their capabilities.[4]

1.5 Establish SLAs around response time, coding and documentation guidelines, or other aspects of multi-team collaboration, shared DoDs. Co-created agile working agreements are much in favor.[5]

[1] Points of Reference could be the Standish Group CHAOS Report, Lean Principles and Wastes, or: https://www.researchgate.net/publication/264001364_Proven_Standards_A_Product_of_Technical_Excellence

[2] There are many definitions of coaching, facilitation, and teaching. We recommend the following associations: International Coaching Federation ICF - https://coachingfederation.org, International Association of Facilitators - https://www.iaf-world.org..

[3] You can find expectations about a coaching agreement at the ICF - https://coachingfederation.org/core-competencies.
EMCC Contracting guide: https://emccuk.org/common/Uploaded%20files/contracting%20guidance.pdf
Expectation from IAF on creating collaborative client relationships: https://www.iaf-world.org/site/professional/core-competencies

[4] There are many sources for formats, exercises and workshops including tastycupcakes.org and liberatingstructures.com.

[5] You can base this on a simple definition of a service level agreement SLA - https://en.wikipedia.org/wiki/Service-level_agreement.

2 - Catalyzing High Performing Technology Organizations

2.1 "Ports and Adapters"/Hexagonal architecture[6], event sourcing, Reactive Architecture Principles, SOLID[7], CUPID[8], Kent Beck's 4 Rules of Simple Design, Code Smells & Refactoring.

2.2 Event storming[9], user story mapping[10], example mapping[11].

2.3 No or low test coverage, monolithic blocks, deprecated components, knowledge buried in code, too large to change, overly complex and tightly coupled.

2.4 Address with: anti-corruption layers, patterns, "elephant carpaccio"[12]; slicing the huge effort into smaller areas of concern, creating outside-in tests.

2.5 Refactor a system for maintainability and present a report of their practice; or demonstrate an exercise during a class. Include metrics or some other way of demonstrating that the refactoring actually improved the code.[13]

2.6 Implement a single-command build, identify blockers, run tests automatically, provide release artifacts, reduce build time, move build logic to code base, remove specific machine dependencies, containerize build system.[14]

2.7 Integration Tests, end-to-end tests, mutation testing, testing databases, UI's, asynchronous calls, hardware components, exploratory testing. [15] [16] [17]

2.8 BDD scenarios with Cucumber[18] [19] [20] [21], Fit and FitNesse test cases, No-Code APIs for acceptance testing. Integration- and external tools like Postman for testing API's.

2.9 Automated embedded or hardware tests, continuous integration of knowledge work beyond software. [22]

[6] https://alistair.cockburn.us/hexagonal-architecture/

[7] SOLID: Single Responsibility, Open-Closed, Liskov-Substitution, Interfaces, Dependency Inversion

[8] CUPID: https://speakerdeck.com/tastapod/cupid-for-joyful-coding

[9] Event Storming: Alberto Brandolini - https://leanpub.com/introducing_eventstorming
https://en.wikipedia.org/wiki/Event_storming or http://ziobrando.blogspot.com/2013/11/introducing-event-storming.html

[10] User Story Mapping: Jeff Patton - https://www.jpattonassociates.com/user-story-mapping/

[11] Example Mapping: Matt Wynne - https://cucumber.io/blog/bdd/example-mapping-introduction/

[12] https://alistair.cockburn.us/wp-content/uploads/2018/02/Elephant-Carpaccio-exercise-instructions.pdf

[13] Clean Code: A Handbook of Agile Software Craftsmanship: Robert C. Martin

[14] Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation: Jez Humble

[15] https://martinfowler.com/bliki/ContractTest.html

[16] Agile Testing: A Practical Guide for Testers and Agile Teams: Lisa Crispin

[17] More Agile Testing: Learning Journeys for the Whole Team: Janet Gregory

[18] Discovery: Explore Behaviour Using Examples: Gáspár Nagy.

[19] Formularion: Express examples using Given/When/Then: Gáspár Nagy.

[20] Behavior-Driven Development with Cucumber: Better Collaboration for Better Software: Paul Rayner.

[21] Specification By Example: Gojko Adzic

[22] Scrum for Hardware: Paolo Sammicheli.

3 - Facilitating environments for a shared understanding

3.1 Self-organized knowledge exchange, fostering understanding and communication, improved adaptability, talent development support. [23]

3.2 break-out sessions, perfection game [24], mirroring, shared UML drawing, specification by example, event storming, swarming, pair/mob/ensemble programming

3.3 Context, hypothesis, planned actions, results, and insights of an experiment the student conducted. Experiments can include a/b testing, long-term measurements, pitches, statistical evaluations.

4 - Guiding Scrum Teams to Learn and Grow

4.1 communities of practice (CoP) [25], training, coding dojos [26], ensemble programming/design sessions, Fitness Function Driven Development [27].

4.2 See 4.1. for examples

5 - Developing Self as an Agile Leader

5.1 none given on purpose [28]

5.2 Approaches to review the own value system and build a bridge towards the Manifesto: Responsibility Process [29], Team Management System, Agile Manifesto Coaching Cards, Moving Motivators [30]. Emphasize the importance of being respected and thus respecting other people's perspectives more.

5.3 Importance of Developer contribution if multiple teams share one Product Owner, "development means the whole value chain", technical aspects of experimentation, Extreme Programming's On-Site Customer. [31]

5.4 Prepare a causal loop diagram [32], value stream map [33], or Kanban [34] system. Discuss batch size, throughput, lead time, cycle time, loops that reinforce or balance system variables.

[23] See 4.1 for references

[24] https://liveingreatness.com/core-protocols/perfection-game/

[25] J. Lave, E. Wenger: Situated Learning: Legitimate Peripheral Participation. Cambridge University Press, Cambridge 1991
E. Wenger: Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press, 1998

[26] https://codingdojo.org/

[27] https://www.thoughtworks.com/insights/articles/fitness-function-driven-development

[28] There is no specific source here. We do not want to limit the Educator or student imagination about the topic of leadership. The intention behind this learning objective is to emphasize leadership on all levels, especially without formal leader positions and roles.

[29] Responsibility Process, Christopher Avery. Self-Leadership.

[30] https://management30.com/practice/moving-motivators/

[31] http://www.extremeprogramming.org/rules/customer.html

[32] Causal loop diagrams: Peter Senge, System Thinking.

[33] Value stream mapping is described in David J. Anderson's work. See also James P. Womack, Daniel T. Jones, and Daniel Roos, "The System that Changed the World".

[34] Kanban, David J. Anderson.

PROGRAM TEAM

CSP-D Program Team Members

- Andreas Schliep
- Axel Berle
- Bill Fairfield
- Björn Jensen
- Falk Kühnel
- Joe Justice
- Martin Salias
- Pascal Gugenberger
- Paul Ballew
- Paul Moore
- Rickard Jones
- Sherman Gomberg
- Rob Myers
- Veronica Ohl