CSPO Learning Objectives
Preamble

This document contains topics for the Certified Scrum Product Owner (CSPO) training course. The purpose of this document is to describe the minimum set of concepts and practices that must be learned by anyone attending a CSPO course. In other words, this document describes the knowledge and understanding that attendees will exit the CSPO training having been exposed to.

How an individual instructor achieves these learning objectives is left to the judgment of the instructor. It is expected that instructors will prepare vastly different course materials and participant experiences while achieving the learning objectives described in this document.

The contents of the document were created by Mike Cohn, Pete Deemer and Roman Pichler.

Scrum Basics

Understand the Scrum Flow, the core components of the Scrum framework, and the Scrum vocabulary

- Teach the Scrum framework; the framework consists of the Scrum team with ScrumMaster, product owner and team; the product backlog as a list of product features and deliverables that is detailed appropriately, emergent, estimated, and prioritized; the sprint with its four meetings and two inspect-and-adapt cycles; the sprint's outcome, a potentially shippable increment of functionality (product increment). Teach that the elements of the framework are mandatory and must not be compromised.
- Teach the properties of sprints: Timeboxed, protected from any changes, max. duration is a calendar month.
- Teach the Definition of Done, the criteria that the product increment must fulfill at the end of a sprint. These usually include high-priority product backlog items implemented as executable software that is thoroughly tested, free of defects, refactored, and adequately documented so that the increment could be shipped.
- Teach that Scrum leverages iterative-incremental software development principles. The software system is usually implemented in form of vertical slices. Each increment extends the previous ones. Each slice should provide value to a user or customer.

Understand the principles/legs of empirical process control

Teach empirical management; empirical management consists of three "legs:"

- Create transparency (see things for what they really are, grasp the situation)

- Inspect (investigate causes)
- Adapt (identify improvement measures).

As a consequence, Scrum does not prescribe how the team goes about its work in the sprint; the team uses inspect-and-adapt to manage and optimize its work. Another consequence is the lack of sophisticated reporting; customers, users and management is expected to attend the sprint review meeting to understand what the project has achieved and help maximize the likelihood of creating a successful product.

Understand the work culture Scrum creates

Characterize the work environment Scrum creates: People collaborate, enjoy their work and create software that benefits its users and customers. A good way to do this is to teach the agile values and the Scrum values; the agile values are described in the Manifesto for Agile Software Development and characterize an agile work environment where people enjoy their work and create software that benefits its users and customers; the five Scrum values are described in "Agile Software Development with Scrum:" Commitment, focus, openness, respect, and courage.

Roles & Responsibilities

Understand the scope of the Product Owner role in detail

- Teach that the Product Owner is responsible for establishing the high-level vision and goals necessary to optimize the return-on-investment for the effort—in other words, what the project will create, for whom, and why.
- Teach that the Product Owner is responsible for obtaining the budget necessary for the work of the team (aka development work) to occur.
- Teach that the Product Owner is responsible for gathering input about what features / functionality should be produced, and understanding the relative value of that functionality.
- Teach that the Product Owner is responsible for creating a list of required features / functionality (the Product Backlog), prioritized based on value, size, risk, and other relevant considerations.
- Teach that the Product Owner is responsible for providing the team with as much context and detail as the team requires to deliver the features / functionality they commit to complete during the Sprint.
- Teach that the Product Owner is required to permit the team to make a reasonable commitment for each Sprint, one which does not sacrifice quality or sustainable pace.
- Teach that the Product Owner is responsible for inspecting the features / functionality produced at the end of each Sprint, and identifying ways to improve its business value.
- Teach that the Product Owner is responsible for evolving the Product Backlog from Sprint to Sprint, in response to insights about opportunities, constraints, and risks.

- Teach that the Product Owner is responsible for deciding when to release software to the market / customer/s.
- Teach that the Product Owner should maintain clear release criteria (target release dates and / or target release scope) and Release Burndown Chart, and share this with the team and stakeholders.
- Teach that the Product Owner is not permitted to disrupt the team or change their commitment during the Sprint, except in the event of a major change in circumstances, in which case the Product Owner may terminate the current Sprint and ask the team to plan a new one.
- Teach that the Product Owner is responsible for making a best effort to provide whatever input or assistance the team requests.
- Teach that the Product Owner is responsible for optimizing the return-on-investment of the product, and for stakeholder management.
- Teach that the Product Owner is an individual and not a committee.
- Teach that the Product Owner should be engaged with the team throughout the Sprint, providing input and feedback as required by the team, but should take care not to disrupt or interfere with the team.
- Teach that the role of the Product Owner is typically played by the customer or customer representative, such as a product manager.

Understand the scope of the ScrumMaster role at a high level

- Teach that the ScrumMaster is responsible for helping the Team remove obstacles and impediments to its ability to deliver high-quality, potentially shippable features / functionality at the end of each Sprint.
- Teach that the ScrumMaster is responsible for protecting the Team from disruption or other threats to their ability to deliver high-quality, potentially shippable features / functionality at the end of each Sprint.
- Teach that the ScrumMaster is responsible for coaching the Team on their practices, and helping them improve their ability to deliver high-quality, potentially shippable features / functionality each Sprint.
- Teach that the ScrumMaster is responsible for facilitating interactions as needed within the Team, and between the Team and Product Owner.
- Teach that the ScrumMaster is responsible for teaching Scrum to the Team, Product Owner, and other people in the organization, and for guiding the skillful use of Scrum.
- Teach that the ScrumMaster is responsible for acting as a change-agent in growing the organization's ability to produce business value early and often, and remove waste.

Understand the scope of the Team role at a high level

- Teach that the team size is 7 people, + or – 2
- Teach that Scrum teams scale by having multiple teams (organized as teams of teams) rather than larger teams.
- Teach that the team should be 100% dedicated to the Sprint, and should sit together
- Teach that the team is self-organizing and empowered to decide how much work to take on in a sprint.

- Teach that the team is collectively responsible for reaching the Sprint goal and meeting the commitment.
- Teach that the team is responsible for working diligently to produce high-quality, potentially shippable functionality each Sprint, without sacrificing sustainable pace of work, or the long-term quality of the work delivered
- Teach that the team should includes all the skills necessary to produce a high-quality increment of potentially shippable product, and that team-members are not constrained in the work they may do by their prior role designation (coder, tester, etc.).

Understand why there is no project manager and no agile product manager

- Teach that Scrum addresses the prior responsibilities of the project manager through the Roles of Product Owner, ScrumMaster, and Team.
- Teach that the delegation of responsibilities (for example, from the Product Owner to a project manager or an "Agile Product Manager") will introduce risk, dysfunction, and waste.

Product Vision

Understand the importance of having the product vision as an overarching goal galvanizing the entire Scrum team

- Teach the importance of having a shared goal; teams need a common goal to work together effectively. The product vision describes the common goal; it aligns the Scrum team members and stakeholders pulling everyone in the same direction.
- Teach the information the vision should contain. The vision should state the customers and users of the software, the needs addressed, and the most important product attributes. It may also compare the product to existing ones and state the revenue model.

Understand the desirable qualities of the vision

Teach that a good vision is:

- Shared. Everyone must buy into it.
- Broad and engaging. The vision should provide guidance for the team while leaving room for creativity and trade-offs.
- Concise. It is often captured on one or two flip chart pages or on a wiki web page.

Understand how the vision can be shaped

Teach techniques for creating the vision, for instance, vision box, elevator statement, press release, magazine review, one page data sheet, personas and scenarios, Kano Model.

Understand the importance of carrying out just enough prep work

Teach that the upfront work in Scrum to create a product vision should be minimized; just enough work to create a vision that gels the team and creates forward momentum; point out the danger of rushing into the project without an overall goal and of procrastinating the project and being trapped in analysis-paralysis; discuss different products and domains that require a different amount of prep work, for instance, embedded software development with hardware and mechanics partners / sub projects vs. a web application created by a small Scrum team; teach the fallacy of trying to predict the future and making fail-safe investment decisions.

Understand the relationship between vision and product roadmap

- Teach that as the product matures and incremental updates are released, the visioning effort usually declines. New product versions still need to have goals to align the Scrum team and stakeholders. Visioning now forms part of creating and updating the product roadmap.
- Teach that a product roadmap is a planning artifact that shows how the product is likely to evolve across releases and product versions, facilitating a dialogue between the Scrum team and the stakeholders.

Estimating

Understand the different estimation levels in Scrum

- Teach that product backlog items and sprint backlog items (tasks) are each estimated, commonly in hours. Teach that product backlog items are estimated. Teach one or more estimating units for product backlog items, such as story points or ideal days.
- Teach the purpose of estimating: The product backlog estimates enable the product owner to prioritize work and the Scrum team to determine the effort remaining to deliver the release thereby tracking the progress and creating a release plan. The task estimates help the team to pull the right number of product backlog items into the sprint allowing the team to make a realistic commitment.

Understand that the accuracy of an estimate is more important than the precision of the estimate

- Teach the difference between accuracy and precision
- Teach that an accurate estimate may be useful even if its precision is less than the product owner would like.
- Teach that an inaccurate (but precise) estimate is of no use.

- Teach that estimates in Scrum are team estimates and should not include any assumptions abut who is going to implement a product backlog items or sign up for a task.

Understand that estimates of size and duration can be done separately

Teach that the team should estimate the size of the effort and then velocity should be used to empirically determine the duration. Size and duration can vary independently.

Understand the impact of pressuring team members to provide low estimates

- Teach that team members will provide low estimates when forced but that they won't finish the work within that time.
- Teach that excessive time pressure leads to cutting quality (which comes back to hurt the project later), low morale, and loss of creativity.

Understand the difference between estimating and committing

- Teach that an estimate is a prediction of the future and always includes some amount of uncertainty, even if that uncertainty isn't stated
- Teach that a commitment is based on an estimate, which must be created first
- Teach the implications of equating estimating with committing

---

The Product Backlog

Understand what the product backlog is (and is not)

- Teach the product backlog essentials: The product backlog is a list of requirements and other deliverables necessary to turn the vision into a successful product. The product backlog must must be **D**etailed appropriately, **E**stimated, **E**mergent, and **P**rioritized (DEEP).
- Teach that the product backlog is not a substitute for a requirements specification. The product backlog evolves and changes constantly; many of its items are coarse-grained and sketchy to start with; they are then progressively decomposed and refined. The emphasis shifts from documentation to conversation, from specifying requirements to having an ongoing dialogue.
- Teach that the product backlog can have different forms and shapes; it may be, for instance, a collection of paper cards or be held electronically as a spreadsheet or in a specialized tool. The product backlog can be flat or structured employing several columns and grouping items into themes, for instance.
- Explain that a product backlog can be augmented as appropriate with other artifacts, for instance, a spreadsheet showing business rules, a Visio diagram

showing a workflow, a user interface prototype or mockup. Teach that these should be only used when necessary and kept as light as possible.

Understand product backlog grooming

- Teach the necessary steps to groom the product backlog such as discovering new requirements, and updating or removing existing ones; prioritizing the backlog; preparing high-priority items for the next sprint planning meeting; estimating items.
- Teach approaches to stocking the product backlog and determining the release scope such as deriving the contents from the product vision.
- Teach that just-enough requirements are identified and described just in time according to their priority throughout the entire project.
- Teach that grooming the product backlog is a collaborative effort shared by all Scrum team members; the product owner leads the efforts; the team members can spend up to 10% of their time on grooming. Grooming often involves customers and users and other stakeholders including marketing and sales.
- Teach that the high-priority items likely to be worked on in the next sprint must be small enough to be fully transformed into a product increment. They should also be clear and testable.
- Talk about at least one technique suitable to capture product backlog items, for instance, user stories.
- Teach how non-functional requirements can be dealt with in the product backlog.

Prioritizing

Understand the importance and benefits of prioritizing the product backlog

- Teach that it is important to prioritize the backlog because the team is expected to work in approximately priority order
- Teach that because it is impossible to perfectly predict a schedule and the contents of a release, there is always the chance that a product will need to be shipped sooner than planned or with fewer features.
- Teach that a prioritized product backlog enables the team to maximize the delivery of value to the customer over a given period of time.
- Teach that prioritizing a product backlog involves both determining what product backlog items will be planned to be in the upcoming release and the general sequence in which they will be built. The sequence should be more approximate the further out in time the features are.

Understand the implications of saying everything is mandatory

- Teach that it is important to prioritize work even on a fixed-scope project for which all features are mandatory.
- Teach that since not all work can be done simultaneously, even a fixed-scope project will still benefit from prioritizing work at the start of each sprint. For

example, even though all features may be considered mandatory, some features should be developed sooner so that they can be shown earlier to users for feedback.
- Teach that it is important to acknowledge that even though a team plans and commits to deliver all features on a fixed-scope project, that does not always happen: Project schedules are sometimes shortened, teams make mistakes, and so on.

Understand who should have input into prioritization decisions

- Teach that it is important for the product owner to incorporate the opinions of many diverse stakeholders.
- Teach that opinions about priorities should be sought from users, customers and other business-side stakeholders, depending on the application.
- Teach that it is also important to seek opinions about priorities from the developers. Such developers are often in the best position to comment on risk and changes over time in the relative cost of product backlog items.

Understand that proper prioritization of a product backlog is based on multiple factors.

Teach that prioritization should be a function of the desirability of the few feature or capability, the amount of learning that will be created by working on the feature, the extent to which risk is reduced by developing the feature, and any changes in the relative cost of the feature over timer.

Understand and know how to apply formal approaches to prioritizing (i.e., beyond just "gut feel" or intuition)

- Teach when such approaches are applicable
- Teach the limitations of formal approaches in general and of any specific approaches described
- Teach that there are many specific techniques that may be used for prioritizing a product backlog.
- Teach at least three formal methods for prioritizing. Examples include non-financial methods such as relative weighting, impact estimation, theme screening, theme scoring, MoSCoW, and others. Financial measures such as net present value, economic value added, return on investment, discounted payback period and others may also be used.

Understand how much latitude to give a team in adjusting the sequence of work.

Teach that although the product owner prioritizes the product backlog, the team should generally be given some latitude in sequencing work into sprints. For example, consider a team that is required to work in strict priority order and has so far planned the top three product backlog items into the sprint. Such a team could find itself in a situation where they are required to work on item four next but it is too big to fit in the sprint. If required to work in strict priority order the

team would be prevented from working on the smaller fifth item. There are many other similar reasons for working slightly out of order; an effective product owner is aware of these situations and works with the team to determine their appropriate application.

Release Management

Understand the goal of release management

- Teach that the primary goal of performing release management activities is to bring the vision to life in the best possible way—to develop a product that benefits the customer/user and the organization creating it.
- Teach that successful release management relies on an evolving and improving understanding of user or customer needs.
- Teach that release management consists of estimating product backlog items, tracking the progress using release burndown charts, and creating a release plan. The latter is not a mandatory part of the Scrum framework.

Understand that planning is adaptive, iterative, and collaborative

- Teach that planning takes place at different levels in Scrum: Product, release, sprint, and day.
- Teach that release planning is spread throughout a project, not just early on. Plans in Scrum are dynamic and change as more information about the customer needs and the product being developed becomes available.
- Teach that release planning activities are carried out by the Scrum team often with the help of stakeholders, for instance, in the sprint review meeting. The product owner ensures that the necessary release management activities take place.

Understand why quality is frozen and the concept of technical debt

- Teach that quality is defined in the definition of done. It is no longer a variable but fixed.
- Teach that quality compromises lead to technical debt, the software becomes brittle and expensive to extend and maintain.
- Teach that short release cycles and frequent software updates require high quality.

Understand why software should be released early and frequently

- Teach that product increments should be released to target customers early and frequently – instead of delivering the finished product in one go
- Teach that early and frequent releases allow incorporating feedback from customers and users, creating a product that meets the needs of its users

Understanding and measuring velocity

- Teach how velocity is defined: The amount of effort earned by the team per sprint; only items delivered according to the definition of done contribute to the velocity.
- Teach that velocity is best observed / measured empirically. This usually requires running at least 2-3 sprints unless the team has worked together on the previous product version.
- Teach how the product owner can help the team optimize its velocity, for instance, by ensuring that high-priority product backlog items are clear, feasible and testable, by jointly detailing high-priority items, and by answering questions in timely manner.

Understand the release burndown chart

- Teach that the release burndown chart is a simple report that creates transparency about the project progress relating time and the effort remaining in the product backlog. It allows the Scrum team and stakeholders to understand how the project is doing.
- Teach that the release burndown chart looks at the past sprints and projects their velocity into the future allowing the Scrum team to make informed decision about managing functionality, time and cost.
- Teach that the release brundown is updated at the end of each sprint when the remaining effort in the product backlog and the velocity are known.
- Teach that the release burndown is no substitute for attending the sprint review meeting to fully understand the project progress and for engaging in dialogue with the Scrum team.

Understand how a release plan can help forecast the future

- Teach that a release plan is a rough forecast, not a commitment and certainly no guarantee. The release plan is no substitute for a project plan; it contains less detail and is updated in each sprint. A traditional project plan no longer exists in Scrum.
- Teach that a release plan is based on the amount of work remaining in the product backlog and the velocity. It looks ahead into the future forecasting delivery date, cost and functionality by anticipating the velocity of the remaining sprints, dependencies to partners and suppliers, and any releases such as alpha and beta releases.

Sprints

Understand the product owner's role in the Scrum meetings

- Teach that the Product Owner should attend and actively participate in the Sprint Planning Meeting, and should come to the meeting with a Product Backlog that is well prepared—in other words, with items at the upper part of the Product Backlog that are clearly thought through, and provide sufficient detail for the team to be able to make a commitment for the Sprint.
- Teach that the Daily Scrum Meeting is the development team's meeting, and is not intended as an update for the Product Owner. Teach that the Product Owner may attend the Daily Scrum Meeting only at the invitation of the team, and may not interfere.

Understand how the Product Owner and Development Team collaborate during the Sprint

- Teach that during the Sprint, the Product Owner should be fully available to answer questions and clarify details that the development team requires in order to deliver their commitment.
- Teach that during the Sprint, the Product Owner should not disrupt or interfere with the development team.
- Teach that during the Sprint, the Sprint Commitment does not change, and that the Product Owner should not attempt to introduce new or changed items during the Sprint.
- Teach that in the event of a significant change in circumstances, the Product Owner may choose to terminate the Sprint and ask the Development Team to begin a new one.
- Teach that if the Product Owner identifies new requirements during the Sprint, they should be placed on the Product Backlog to be worked on in a future Sprint.
- Teach that the Product Owner and development team should together decide the Sprint length, and share the characteristics of longer and shorter Sprints
- Teach that once the Sprint begins, its duration is never extended.
- Teach that the Sprint commitment does not equal a guarantee

Understand what team commitment means

Understand why sprints are timeboxed and protected

Understand the concept of sustainable pace

References

Cockburn, Alistair. Writing Effective Use Cases. Addison-Wesley. 2001.

Cohn, Mike. User Stories Applied: For Agile Software Development. Addison-Wesley. 2004.

Cohn, Mike. Agile Estimating and Planning. Prentice Hall. 2005.

Cohn, Mike. Succeeding with Agile: Software Development using Scrum. Addison-Wesley. 2009.

Deemer, Pete; Benefield, Gabrielle; Larman, Craig; Vodde, Bas. The Scrum Primer. September 2006.

Pichler, Roman. Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley. 2010.

Schwaber, Ken. Scrum Guide. Scrum Alliance. June 2009.

Schwaber, Ken. The Enterprise and Scrum. Microsoft Press. 2007.

Schwaber, Ken. Agile Project Management with Scrum. Microsoft Press. 2004.

Schwaber, Ken and Mike Beedle. Agile Software Development with SCRUM. Prentice Hall. 2002.