Empiricism & Values

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

LEARNING THROUGH EVIDENCE

Transparency See reality → **Inspection** Examine evidence → **Adaptation** Change approach → **Learning** Repeat

01 Empiricism and Scrum values

Complexity and empirical learning

- **Scrum** provides a lightweight framework for complex problems. Requirements, technology and working relationships create uncertainty; detailed initial plans cannot remove it.
- **Empiricism** bases decisions on observed experience. **Lean thinking** reduces waste. Short cycles of usable results expose assumptions and support better decisions.
- **Transparency, inspection and adaptation** work together: make reality visible, examine progress and change course when evidence demands it.
- **Iterative** work revisits and improves a solution; **incremental** work adds usable capability. Together they limit exposure and create frequent learning opportunities.

Values as observable behavior

- **Commitment** means pursuing shared goals and supporting colleagues. **Focus** concentrates effort on the Sprint's objective and the most valuable work.
- **Openness** makes work, uncertainty and problems discussable. **Respect** includes listening to new or dissenting voices. **Courage** enables difficult conversations and sound quality decisions.
- **Trust** grows when behavior consistently reflects these values. Ignoring ideas, hiding unfinished work or punishing bad news weakens the evidence available for decisions.
- **Self-management** operates within clear goals, timeboxes and quality expectations. Autonomy includes professional accountability and collaboration.

Make the real product state visible

- The **Product Backlog**, **Sprint Backlog** and **Increment** expose work and value. Their commitments are the **Product Goal**, **Sprint Goal** and **Definition of Done**, respectively.
- Share inspectable results, remaining work, dependencies, risks and forecast assumptions. A green status report cannot compensate for unusable or unintegrated work.
- When perceived progress exceeds actual progress, disclose the gap promptly. Make unfinished work visible and revise forecasts using the effort required to reach Done.

Leadership & Coaching

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

ENABLE EFFECTIVE DECISIONS

Observe — Understand context → **Explore** — Ask and listen → **Enable** — Teach or facilitate → **Inspect** — Learn from action

02 Scrum Master leadership and coaching

Lead through service

- The **Scrum Master** is accountable for establishing Scrum and enabling the Scrum Team's effectiveness. Teach the framework and help people improve their practices within it.
- **Servant leadership** develops others' ability to decide and act. Support self-management, productive events and impediment removal without becoming the team's task dispatcher.
- Coach the **Product Owner** on goals, backlog management and empirical planning; coach **Developers** on collaboration and quality; help the **organization** remove barriers to value.
- **Management** supplies an enabling environment, resources and support. Renaming existing meetings and job titles alone leaves the underlying behavior unchanged.

Choose an intervention deliberately

- **Observe before prescribing**: clarify the desired outcome, gather perspectives and explore causes. A long Daily Scrum alone does not justify splitting the team.
- **Coaching** uses questions to build awareness and ownership. **Teaching** explains a missing concept. **Facilitation** helps a group collaborate toward a useful outcome.
- Use one-to-one conversations, group learning or focused training as needed. Help people understand an event's purpose before imposing a meeting technique.
- Encourage **direct communication** between Product Owner, Developers and stakeholders. A permanent intermediary increases delay and dependence on the Scrum Master.

Turn disagreement into learning

- **Conflict** can reveal important differences in assumptions. Help the people involved discuss the issue, its effect on the goal and a way forward they can own.
- Create **psychological safety** for concerns about dominant experts, ignored ideas or repeatedly reopened decisions. Explore how decisions are made and how dissent is heard.
- For Product Owner–Developer tension, examine availability, changing needs and emerging technical knowledge together. Clarify decision rights and agree better collaboration practices.
- When emotions overwhelm a discussion, a short **pause** can restore focus. Return to evidence, shared goals and valuable next steps; do not automatically escalate routine disagreement.
- Use organizational authority and appropriate employment processes for personnel decisions. Scrum accountabilities do not themselves grant hiring or dismissal powers.

Leadership & Coaching / Product & Value

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

PLAN TOWARD VALUE

Goal Desired outcome → **Backlog** Ordered options → **Increment** Usable result → **Feedback** Adapt investment

Remove barriers and build capability

- **Cause impediments to be removed**: make their impact visible, prioritize them and involve people with the authority or expertise to act.
- Help the team address problems it can solve; act on wider organizational barriers with management. Solving everything personally creates a bottleneck and weakens learning.
- Inspect an expanding impediment list for recurring causes, ownership and follow-through. Track whether action improves outcomes, rather than merely counting closed issues.
- Treat **adoption** as organizational change: teach customers and stakeholders, support experiments and address incentives or structures that obstruct collaboration. A mature team still needs Scrum Master accountability.

03 Product ownership, value and forecasting

One accountable value decision maker

- The **Product Owner** maximizes product value and is accountable for effective Product Backlog management. The organization must respect the Product Owner's decisions.
- One product has **one Product Owner**, even with multiple teams. The Product Owner may delegate work while retaining accountability; a committee cannot share that accountability.
- Separate products need clear product boundaries and ownership. One person may serve several products when availability and decision quality remain adequate.
- Measure **customer and business outcomes**, such as satisfaction, adoption or useful benefits. More story points or busier people do not establish greater value.
- Build relationships with users and stakeholders throughout development. Stay available for clarification; effective engagement is judged through understanding and results, not attendance percentages.

Order and refine toward a goal

- The **Product Goal** describes a future product state. The **Product Backlog** is an emergent, ordered list of improvements needed to move toward it.
- Order work using value, risk, dependencies and learning needs. The Product Owner remains accountable for ordering; Developers contribute technical constraints and trade-offs.
- **Refinement** continuously breaks down and clarifies items. Product Owner and Developers collaborate during Sprints; Scrum prescribes neither a refinement event nor a fixed capacity percentage.

Product & Value / Goals & Events

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

PLAN TOWARD VALUE

Goal — Desired outcome → **Backlog** — Ordered options → **Increment** — Usable result → **Feedback** — Adapt investment

- The **Developers doing the work size it**. Story points, Fibonacci scales, user stories and use cases are optional practices, rather than framework requirements.
- Items should be understood well enough to select and complete within a Sprint. An optional **Definition of Ready** should support discussion without becoming a rigid handoff or approval barrier.

Forecast with evidence and uncertainty

- A **forecast** expresses what appears feasible under stated assumptions. Update it when scope, capacity, quality, dependencies or knowledge changes.
- **Velocity** is an optional, team-specific measure of completed work used for planning. It is unsuitable for ranking teams or promising a forced percentage increase.
- **Burndown** shows remaining work; **burnup** shows completed work and can expose scope changes. Trend projections depend on the assumptions behind the data.
- Revisit **funding and expected value** frequently. Several Sprints may be funded together while each still creates a usable result that supports further investment decisions.
- When a delivery date becomes unlikely, expose the gap and discuss scope, ordering, timing and investment with the Product Owner and stakeholders. Adding people does not guarantee proportional speed or value.

04 Sprint goals, events and adaptation

A steady cadence with clear boundaries

- A **Sprint** has a fixed length of one month or less. The next Sprint starts immediately; planning, reviews and improvement happen inside Sprints.
- Choose a cadence that supports useful delivery and timely feedback at acceptable risk. Shorter future Sprints can shorten learning loops; avoid extending an active Sprint to finish its forecast.
- A **timebox** sets a maximum duration. For a one-month Sprint: Planning up to **8 hours**, Review up to **4 hours**, Retrospective up to **3 hours**; these events are usually shorter for shorter Sprints.
- Only the **Product Owner** may cancel a Sprint, when its goal becomes obsolete. Difficulty, unfinished scope or an impediment is not by itself a reason to cancel.
- Scrum defines no special **Sprint 0**, hardening or release Sprint. Develop infrastructure and architecture alongside usable functionality, including from the first Sprint.

Goals & Events

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

A SPRINT'S LEARNING CYCLE

Planning — Form a goal → **Daily Scrum** — Adapt the plan → **Review** — Inspect outcomes → **Retrospective** — Improve

Plan around a coherent objective

- **Sprint Planning** addresses why the Sprint is valuable, what can be Done and how to deliver it. The whole Scrum Team collaborates; advisors may contribute expertise.
- The **Sprint Goal** is a single objective created during Planning. It provides coherence and flexibility in the exact work needed to achieve it.
- Developers select a feasible amount of work with the Product Owner using past performance, upcoming capacity and the Definition of Done. They own the implementation plan.
- The **Sprint Backlog** combines the goal, selected items and delivery plan. Developers update it as learning occurs; it is a living plan rather than a fixed scope promise.
- When details are unclear, refine enough to form a credible initial plan and continue learning. Keep Planning within its timebox and improve preparation through the Retrospective.

Inspect progress every working day

- The **Daily Scrum** is a **15-minute** event for Developers to inspect progress toward the Sprint Goal and adapt the plan. Use a consistent time and place, including a suitable remote setup.
- Developers choose the structure. Three questions, reporting to a manager, individual applause and updating a particular chart are not its purpose.
- Move detailed technical discussions outside the event. Coach dominant speakers, lateness or remote friction by helping Developers improve their own coordination.
- Reducing frequency delays discovery of impediments and weakens alignment. Address the logistical problem while preserving daily inspection and adaptation.
- Replan throughout the day as needed. If too much work was selected, collaborate with the Product Owner to renegotiate scope while protecting the goal and quality. Route new stakeholder requests through the Product Owner.

Inspect outcomes with stakeholders

- The **Sprint Review** is a working session to inspect results and progress toward the Product Goal, consider environmental changes and decide useful adaptations.
- Bring key stakeholders into a conversation about the usable product, new information and future choices. A demonstration alone does not provide the full learning opportunity.
- The Review is **not a release gate**. Done Increments may be delivered during the Sprint; user feedback can inform the Product Backlog without a pause between Sprints.
- Stakeholder collaboration continues outside Reviews. Surprises about suitability, cost or progress often signal weak transparency or inadequate engagement.

Goals & Events / Done & Quality

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

A SHARED QUALITY BOUNDARY

- **Definition of Done** — Quality measures
- **Delivery** — Build and verify
- **Increment** — Integrated and usable
- **Transparency** — Honest product state

Improve quality and effectiveness

- The **Sprint Retrospective** concludes the Sprint. The Scrum Team, including the Product Owner, examines interactions, processes, tools, assumptions and the Definition of Done.
- Identify the most helpful improvements and act promptly. Important changes may enter the next Sprint Backlog; inspect whether they actually improve results.
- If Retrospectives feel useless, improve their safety, focus and follow-through. Omitting the event removes a deliberate learning opportunity.
- Address urgent quality or collaboration issues when discovered. The next Retrospective is an opportunity to learn further, rather than a reason to defer action.

05 Done, product quality and technical debt

Done is a shared quality boundary

- The **Definition of Done** describes the quality state required of the Increment. It creates a common understanding of completion and guides planning and remaining work.
- The Scrum Team creates an appropriate definition when no organizational standard exists. Organizational standards are a minimum; Developers conform to the definition.
- A usable **Increment** builds on earlier Increments and works with them. Multiple Increments may be created in a Sprint; all require verification and integration.
- An item that fails the Definition of Done is **not part of the Increment** and returns to the Product Backlog for consideration. Partial completion cannot be declared Done through stakeholder acceptance.
- Do not lower quality to fit a date or disguise omitted tests. Explore the underlying constraint, reduce selected scope and improve the capability to meet quality expectations.

Make quality concerns actionable

- **Non-functional requirements** include security, performance, scalability and maintainability. Make relevant expectations explicit alongside functional behavior.
- Use **Product Backlog items** for specific improvements and the **Definition of Done** for recurring quality measures. Involve specialists when needed while retaining team accountability.
- When a new risk appears, share it promptly, assess the impact on existing and future work and coordinate action. Do not hide it in a separate security or performance phase.

Done & Quality / Teams & Scaling

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

A SHARED QUALITY BOUNDARY

- **Definition of Done** — Quality measures
- **Delivery** — Build and verify
- **Increment** — Integrated and usable
- **Transparency** — Honest product state

- Build required testing and acceptance checks into the delivery approach. Deferring essential **UAT**, regression or integration testing obscures readiness and lengthens feedback loops.
- If documentation is required for Done, the team must produce it even when its usual writer is absent. A specialist's absence exposes a skill dependency; it does not waive the quality requirement.

Expose debt and restore honest forecasts

- **Technical debt** creates future costs through weaknesses or shortcuts in the product. Its effects can include slower change, instability, rework and interruptions.
- Shortcuts can inflate apparent throughput while making the product less usable. High velocity cannot compensate for missing quality or unusable results.
- Make debt and its consequences visible to the Product Owner. Discuss remediation and trade-offs; required work to meet Done cannot simply be deferred while claiming completion.
- A proposed **hardening Sprint** signals a need to examine the definition, compliance with it and delivery capability. Integrate quality into ordinary Sprints rather than accumulating hidden release work.
- For recovery, expose unfinished work, estimate what remains and revise the forecast. Prefer a trustworthy usable product state over continuing to build on misleading progress figures.

06 Team effectiveness and scaling Scrum

Build teams capable of end-to-end value

- **Cross-functionality** means the team collectively has the skills needed to create value. It does not require every individual to master every discipline.
- A **feature team** can deliver across technical layers. Component silos create handoffs and integration dependencies that can obscure whole-product progress.
- Help people organize into cohesive teams within clear product and skill boundaries. Scrum Teams have no internal subteams or hierarchy and are typically **10 or fewer people**, including all accountabilities.
- Formation, membership changes and adding teams can temporarily reduce productivity through learning and coordination. Stable collaboration helps; reorganize when justified by evidence.
- Encourage knowledge sharing, pairing and broader skills. Team responsibility for the result persists even when individuals specialize.

Teams & Scaling

Professional Scrum Master II

KNOWLEDGE CHEATSHEET

ONE PRODUCT, ALIGNED TEAMS

- **Direction** — One Product Goal
- **Ownership** — One Product Owner
- **Work** — One Product Backlog
- **Quality** — Shared Definition of Done

Keep the product whole

- Teams working on the same product share **one Product Goal, one Product Backlog and one Product Owner**. They mutually define and comply with the same Definition of Done.
- Developers coordinate selection with the Product Owner and other teams. Avoid allocation rules based on equal item counts, velocity rankings or separate team priorities.
- Teams coordinate their work to produce integrated, usable results each Sprint. Separate branches awaiting a later integration phase do not demonstrate a Done product.
- The Scrum Master enables coordination and removes barriers. Teams retain ownership of delivery plans; scaling does not turn the Scrum Master into a central task scheduler.

Reduce dependencies before adding control

- **Dependencies** restrict the flow of value. Inspect them at product level; individually on-track component plans can still produce a late integrated product.
- For a scarce specialist, facilitate a joint plan: share knowledge, build skills and discuss backlog ordering or additional expertise with the Product Owner.
- Minimize fragmented assignments and context switching. A person split across teams is not equivalent to dedicated capacity for each team.
- Tools and scaling techniques may support collaboration, but cannot guarantee it. Add practices that fit the situation while preserving Scrum's accountabilities, events, artifacts and usable outcomes.