Scrum Checklist of Team 10

Date: 01.11.2022

Part I -- How Is My Product Owner Doing?

Scrum Masters improve Product Owner effectiveness in many ways.

☑ Regardless of the number of teams, is there one (and only one[1]) Product Owner for the product?

☑ Is the Product Backlog prioritized according to his/her latest thinking?

☑ Are newly discovered requirements and desirements being captured in the Product Backlog? Remember: the backlog is *emergent*.

☑ Is the Product Backlog a manageable size? To maintain a manageable number of items, keep things more granular towards the top, with general epics at the bottom. It's counterproductive to overanalyze too far past the top of the Product Backlog. Plans will change in an ongoing conversation between the developing product and the users/customers.

△ Could any requirements (especially those near the top of the Product Backlog) be better expressed as independent, negotiable, valuable, estimable, small, and testable user stories[2]?

☒ Have you educated your Product Owner about *technical debt* and how to avoid it? One piece of the puzzle may be to write automated test and refactoring into the definition of "done" for each backlog item.

☒ Is the backlog an *information radiator*, immediately visible to all concerned?

☒ If you're using an automated tool for backlog management, does everyone know how to use it easily? Sadly, automated management tools usually become anti-collaborative *information refrigerators*.

Can you help radiate information by teaching how to create big visible charts?

☑ Have you helped your Product Owner organize backlog items into appropriate releases or priority groups?

☒ Does everyone know whether the release plan still matches reality? You might try showing everyone Product/Release Burndown Charts[3] after the items have been acknowledged as "done" during every Sprint Review Meeting. Charts showing both the rate of PBIs actually completed and new ones added allow early discovery of scope/schedule drift.

☒ Did your Product Owner adjust any release forecasts after the last Sprint Review Meeting? If you want to ship a properly tested product by a fixed date, you must constantly reconsider scope. This probably requires deferring some work for future releases as more important work is discovered. Some products can be released every Sprint, or multiple times per Sprint.

Part II -- How Is My Team Doing?

While the Scrum Master are encouraged to lead by the example of collaborating with team members on their work, there is a risk of getting lost in technical tasks. Consider your primary responsibilities to the team:

- ☑ Is your team in the state of *flow*? Some characteristics of this state[4]:
  - Clear goals (expectations and rules are discernible and goals are attainable, aligning appropriately with one's skill set and abilities).
  - Concentration and focus, a high degree of concentration on a limited field of attention.
  - A loss of the feeling of self-consciousness, the merging of action and awareness.
  - Direct and immediate feedback (successes and failures in the course of the activity are apparent, so that behavior can be adjusted as needed).
  - Balance between ability level and challenge (the activity is neither too easy nor too difficult).
  - A sense of personal control over the situation or activity.
  - The activity is intrinsically rewarding, so there is an effortlessness of action.

- ☑ Do team members seem to like each other, goof off together, and celebrate each other's success?
- ☑ Do team members hold each other accountable to high standards, and challenge each other to grow?
- ☒ Are there issues and opportunities the team isn't discussing because they're too uncomfortable?[5]
- ☒ Have you tried a variety of formats and locations for Sprint Retrospectives?[6]
- ☑ Has the team kept focus on Sprint goals? Consider a mid-Sprint checkup to reassess the Sprint Plan.
- △ Does the Sprint taskboard reflect what the team is actually doing? Beware the "dark matter" of undisclosed tasks and tasks bigger than one day's work. Work not related to the Sprint is an impediment to the Sprint.

- ☑ Does your team have 3-9 people with a sufficient mix of skills to build a potentially shippable product increment?

- ☑ Is your team's taskboard up to date?
- ☒ Are the team self-management artifacts visible to the team, convenient for the team to use?
- ☒ Are these artifacts adequately protected from meddlers? Excess scrutiny of daily activity by people outside the team may impede team internal transparency and self management.

- ☑ Do team members volunteer for tasks? On my first Scrum team we felt like volunteers who also got paid. If it doesn't feel like that, something's gone wrong somewhere.

- ☒ Has the need for technical debt repayment been made explicit in the definition of *done*, gradually making the code a more pleasant place to work?

- ☒ Are team members leaving their job titles at the door of the team room, collectively responsible for all aspects of agreed work (testing, user documentation, etc.)?

Part III -- How Are Our Engineering Practices Doing?

☐ Does your system in development have a "push to test" button allowing anyone (same team or different team) to conveniently detect when they've caused a regression failure (broken previously-working functionality)? Typically this is achieved through the xUnit framework (JUnit, NUnit, etc.).

☐ Do you have an appropriate balance of automated end-to-end system tests (a.k.a. "functional tests") and automated unit tests?

☐ Is the team writing both system tests and unit tests in the same language as the system they're developing? Collaboration is not enhanced by proprietary scripting languages or capture playback tools that only a subset of the team knows how to maintain.

☐ Has your team discovered the useful gray area between system tests and unit tests?

☐ Does a continuous integration[7] server automatically sound an alarm when someone causes a regression failure? Can this feedback loop be reduced to hours or minutes? ("Daily builds are for wimps." -- Kent Beck)

☐ Do *all* tests roll up into the continuous integration server result?

☐ Have team members discovered the joy of continuous design and constant refactoring[8], as an alternative to Big Up Front Design? Refactoring has a strict definition: changing internal structure without changing external behavior. Refactoring should occur continuously, or at least several times per hour. Refactor whenever there is duplicate code, complex conditional logic (visible by excess indenting or long methods), poorly named identifiers, excessive coupling between objects, etc. Refactoring with confidence is only possible with automated test coverage. Neglecting refactoring makes it hard to change the product in the future, especially since it's hard to find good developers willing to work on bad code.

☐ Does your definition of "done" for each Product Backlog Item include full automated test coverage and refactoring? Learning Test Driven Development (TDD) increases the probability of achieving this.

☐ Are team members pair programming or mob programming most of the time? This can dramatically increase code maintainability and reduce defect rates. It challenges people's boundaries and sometimes seems to take longer (if we measure by lines of code rather than shippable functionality). Lead by example by initiating paired workdays or mobbing sessions with team members. Some of them will start to prefer working this way.

Part IV -- How Is the Organization Doing?

☐ Is the appropriate amount of inter-team communication happening? "Scrum of Scrums" is only one way to achieve this, and rarely the best.[9]

☐ Are teams able to produce working features, even spanning architectural boundaries?[10]

☐ Are you conducting an *Overall Retrospective*[11] to resolve cross-team, organizational and systemic problems within the organization?

☐ When appropriate, are the organizational impediments pasted to the wall of the development director's office? Can the cost be quantified in dollars, lost time to market, lost quality, or lost customer opportunities? (But learn from Ken Schwaber's mistakes: "A dead Scrum Master is a useless Scrum Master."[12])

☐ Is your organization one of the few with career paths compatible with the collective goals of your teams? Answer "no" if there's a career incentive[13] to do programming or architecture work at the expense of testing, test automation, or user documentation.

☐ Has your organization been recognized by the trade press or other independent sources as one of the best places to work, or a leader in your industry?

☐ Are you creating a *learning organization*?

Conclusion

If you can check off most of these items and still have time left during the day, I'd like to hear from you.

There's no canned formula for creating human ingenuity. This paper lists points which may, or may not, help in your situation.

Once you start to realize what you could do to make a difference, you may find yourself afraid to do it. This is a sign you're on the right track.

MARKING INSTRUCTIONS

√   (for "doing well")

Δ   (for "could be improved and I know how to start")

?   (for "could be improved, but how?")

X   (for "not applicable" or "would provide no benefit")