An Example Checklist for ScrumMasters

Michael James
(mj4scrum@gmail.com)
14 September 2007
(Revised 2 Feb 2016)

A Full Time Facilitator?

An adequate ScrumMaster can handle two or three teams at a time. If you're content to limit your role to organizing meetings, enforcing timeboxes, and responding to the impediments people explicitly report, you can get by with part time attention to this role. The team will probably still exceed the baseline, pre-Scrum expectation at your organization, and probably nothing catastrophic will happen.

But if you can envision a team that has a great time accomplishing things no one previously thought possible, within a transformed organization, consider being a *great* ScrumMaster.

A great ScrumMaster can handle *one* team at a time.

We recommend one dedicated ScrumMaster per team of about seven when starting out.

If you haven't discovered all the work there is to do, tune in to your Product Owner, your team, your team's engineering practices, and the organization outside your team. While there's no single prescription for everyone, I've outlined typical things I've seen ScrumMasters overlook. Please mark each box with √, Δ, ?, or N/A, as described on the last page.

Part I -- How Is My Product Owner Doing?

ScrumMasters improve Product Owner effectiveness by helping them find ways to maintain the Product Backlog and release plan. (Note that the Product Owner is the one responsible for the prioritized backlog.)

- ☐ Is the Product Backlog prioritized according to his/her latest thinking?
- ☐ Are requirements and desirements from all stakeholders 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. Your requirements will change in an ongoing conversation between the developing product and the stakeholders/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[1]?
- ☐ 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 stakeholders?
- ☐ If you're using an automated tool for backlog management, does everyone know how to use it easily? Automated management tools introduce the danger of becoming *information refrigerators* without active radiation from the ScrumMaster.

---

[1] http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

- [ ] Can you help radiate information by showing everyone printouts?
- [ ] Can you help radiate information by creating 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[2] 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 the release plan after the last Sprint Review Meeting? The minority of Product Owners who ship adequately tested products on time *re-plan* the release every Sprint. This probably requires deferring some work for future releases as more important work is discovered.

Part II -- How Is My Team Doing?

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

- [ ] Is your team in the state of *flow*? Some characteristics of this state[3]:
  - 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/opportunities the team isn't discussing because they're too uncomfortable?[4]
- [ ] Have you tried a variety of formats and locations for Sprint Retrospective Meetings?[5]
- [ ] Has the team kept focus on Sprint goals? Perhaps you should conduct a mid-Sprint checkup to re-review the acceptance criteria of the Product Backlog Items committed for this Sprint.

---

[2] Mike Cohn, *Agile Estimation and Planning*. (2005).

[3] Mihaly Csikszentmihalyi, *Flow: The Psychology of Optimal Experience* (1990).

[4] Marshall Rosenberg, *Nonviolent Communication: A Language of Life: Life-Changing Tools for Healthy Relationships* (2003). Also consider enlisting a professional facilitator who can make uncomfortable conversations more comfortable.

[5] Derby/Larson *Agile Retrospectives: Making Good Teams Great* (2006).

- [ ] 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. Tasks not related to Sprint commitments are impediments to those commitments.

- [ ] 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?

- [ ] 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[6]?

- [ ] 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 several times per hour, 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

---

[6] http://blogs.collab.net/agile/junit-is-not-just-for-unit-testing-anymore

[7] http://www.martinfowler.com/articles/continuousIntegration.html

[8] Martin Fowler, *Refactoring: Improving the Design of Existing Code* (1999).

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 most of the time? Pair programming may dramatically increase code maintainability and reduce bug 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 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 independently able to produce working features, even spanning architectural boundaries?[10]

- [ ] Are your ScrumMasters meeting with each other, working the organizational impediments list?

- [ ] 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 ScrumMaster is a useless ScrumMaster."[11])

- [ ] 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[12] 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.

---

[9] See http://less.works/less/framework/coordination-and-integration.html for alternatives.

[10] http://FeatureTeamPrimer.org/

[11] Ken Schwaber, *Agile Project Management with Scrum* (2004)

[12] Alfie Kohn, *Punished By Rewards: The Trouble with Gold Stars, Incentive Plans, A's, Praise, and Other Bribes* (1999)

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

Organizational Impediment Form

Surface issue:

Root cause (Use five times "Why?"):

Business Impact:

Emotional Impact:

Clear, actionable request:

INSTRUCTIONS

If you have received this checklist as a training assignment and your current (or most recent) employer has been attempting anything like Scrum, please apply this to what you've seen there. Mark each item with one of the following:

- √ (for "doing well")
- Δ (for "could be improved and I know how to start")
- ? (for "could be improved, but how?")
- N/A (for "not applicable" or "would provide no benefit")

Or, if your current (or most recent) employer has not been attempting anything like Scrum, mark each item with one of the following:

- √ (for "doing well" or "would be easy to do well")
- Δ (for "would be a challenge and I know how to start")
- ? (for "would be a challenge and I don't know how to start")
- N/A (for "not applicable" or "would provide no benefit")

When all items are marked, declare 2-6 organizational impediments on the attached Organizational Impediment Forms, whether or not they're derived from this checklist. Choose impediments you have at least 1% hope of changing.