



**Michael James**  
Michael James is a software process mentor and Certified Scrum Trainer, focusing on the engineering practices that enable Agile project management. Having worked in the software industry for more than 20 years as a software developer (formerly “architect”), he has experience in automated testing that predates the Extreme Programming movement; formal, phased, high-ceremony processes based on DOD-STD-2167A; chaotic non-processes of the dot-com era; and Agile processes including Scrum and XP.

# A ScrumMaster’s Checklist

Posted by Michael James on August 13, 2007 at 5:09 a.m.

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 chances are nothing catastrophic will happen.

But if you can envision a *hyperproductive* team -- a team that has a great time accomplishing things no one else can -- 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, especially when starting out.

If you haven’t discovered all the work there is to do, tune in to your Product Owner, your team and its engineering practices, and the organization outside your team. While there’s no single prescription, I’ve outlined some things I’ve seen ScrumMasters overlook.

1. How is my Product Owner doing?

You improve the Product Owner’s effectiveness by helping maintain the Product Backlog and release plan. (Note that only the Product Owner may prioritize the backlog.)

- Is the Product Backlog prioritized according to his/her latest thinking?
- Are all the stakeholders’ requirements and desires for the product captured in the 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.

**danube**

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

- 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?
  - Have you educated your Product Owner about technical debt<sup>1</sup> and how to avoid it? One piece of the puzzle may be adding automated testing and refactoring to the definition of “done” for each backlog item.
  - Is the backlog an *information radiator*, highly 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<sup>2</sup> without active radiation from the ScrumMaster.
  - Are you working with the tool supplier to maximize its potential or to change it to serve you better?
  - Can you help radiate by showing everyone printouts?
  - Can you help radiate by creating big visible charts<sup>3</sup>?
  - Have you helped your Product Owner organize backlog items into appropriate releases (or front burner, back burner, fridge)?
  - Do all stakeholders (including the team) know whether the release plan still matches reality, based on the current velocity (story points per Sprint)?
  - 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, usually deferring some work for future releases as more important work is discovered. You might try showing everyone the Mike Cohn-style Product/Release Burndown Charts after the items have been acknowledged as “done” during every Sprint Review Meeting. This allows early discovery of scope/schedule drift.
2. How is my team doing?

---

<sup>1</sup> [http://danube.com/blog/kanemar/technical debt and the death of design part 1.html](http://danube.com/blog/kanemar/technical%20debt%20and%20the%20death%20of%20design%20part%201.html)

<sup>2</sup> <http://c2.com/cgi/wiki?InformationRefrigerator>

<sup>3</sup> <http://www.xprogramming.com/xpmag/BigVisibleCharts.htm>



- Are team members spending some of their time in the state of flow<sup>4</sup>? Some characteristics of this state (from *Flow: The Psychology of Optimal Experience* by Mihaly Csikszentmihalyi) include:
  - Clear goals, expectations and rules that are discernible and goals that are attainable and align 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.
  - Distorted sense of time in which one's subjective experience of time is altered.
  - Direct and immediate feedback, in which 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, so that 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 to high standards, and challenge each other to grow?
- Are there issues/opportunities the team isn't discussing because they're too uncomfortable? If so, take a look at *Crucial Conversations: Tools for Talking When Stakes are High*. Or consider enlisting a professional facilitator<sup>5</sup> who can make "uncomfortable conversations" more comfortable.
- Have you tried a variety of formats and locations for Sprint Retrospective Meetings? See *Agile Retrospectives: Making Good Teams Great* (Esther Derby/Diana Larsen) for some ideas.
- Has the team kept focus on acceptance criteria? Perhaps you should conduct a mid-Sprint checkup to re-review the acceptance criteria of the product backlog items committed for this Sprint.



Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

---

<sup>4</sup> [http://en.wikipedia.org/wiki/Flow\\_%28psychology%29](http://en.wikipedia.org/wiki/Flow_%28psychology%29)

<sup>5</sup> <http://www.compassionate-coaching.com/Facilitation.htm>

Will the current Sprint Task list satisfy the acceptance criteria of each committed product backlog item?

- Does the Sprint Task list reflect what the team is actually doing?
- Are your team's task estimates and/or your taskboard up to date?
- Are the team self-management artifacts (taskboard, Sprint Burndown Chart, etc.) visible to the team, convenient for the team to use?
- Are these artifacts adequately protected from micromanagers?
- Do team members *volunteer* for tasks?
- Are technical debt repayment items<sup>6</sup> (issues that sap your team's velocity) captured in the backlog, or otherwise communicated with the Product Owner?
- Are team members checking their job titles at the door of the team room?
- Does the entire team consider itself collectively responsible for testing, user documentation, etc.?
- Is management measuring the team by collective success?

3. How are our engineering practices doing?

- Does your system in development have a “push to test” button so that anyone (same team or different team) can conveniently detect when they've broken it? Typically, this is achieved through the xUnit framework (JUnit, NUnit, etc.).
- Do you have an appropriate balance<sup>7</sup> between automated end-to-end system tests (a.k.a. “functional tests”) and automated unit tests?
- Is the team writing both system “functional” tests and unit tests in the same language as the system they're developing (rather than using proprietary scripting languages or capture playback tools only a subset of the team knows how to maintain)? It's possible<sup>8</sup>.
- Has your team discovered the useful gray area between system tests and unit tests?

---

<sup>6</sup> [http://danube.com/blog/michaeljames/how\\_to\\_survive\\_technical\\_debt%23comment-1214](http://danube.com/blog/michaeljames/how_to_survive_technical_debt%23comment-1214)

<sup>7</sup> [http://danube.com/blog/michaeljames/mock\\_objects\\_considered\\_insufficiently\\_harmful](http://danube.com/blog/michaeljames/mock_objects_considered_insufficiently_harmful)

<sup>8</sup> [http://danube.com/blog/michaeljames/junit\\_is\\_not\\_just\\_for\\_unit\\_testing\\_anymore](http://danube.com/blog/michaeljames/junit_is_not_just_for_unit_testing_anymore)

The Danube logo consists of the word "danube" in a white, lowercase, sans-serif font, centered within a solid blue rectangular background.

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

- Does a continuous integration<sup>9</sup> server automatically sound an alarm within an hour (or minutes) of someone causing a regression failure? (“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<sup>10</sup> as an alternative to Big Up Front Design? Refactoring has a strict definition: changing the internal structure of a system without changing its 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. Holes in test coverage and deferred refactoring are cancers called technical debt.
  - Does your definition of “done” (acceptance criteria) for each functional Product Backlog Item include full automated test coverage and refactoring? I’ve never seen or heard of a hyperproductive team that wasn’t doing the eXtreme Programming practices (as described by Kent Beck, Ron Jeffries, etc.).
  - Are team members pair programming most of the time? Pair programming dramatically increases code maintainability and reduces bug rates. But it challenges people’s boundaries and sometimes seems to take longer (if we put quantity over quality). Rather than force people to do this, lead by example by initiating paired workdays with team members. Some of them will start to prefer working this way.
4. How is the organization doing?
- Is the appropriate amount of inter-team communication happening? Scrum-of-Scrums is only one way to achieve this.
  - Are your ScrumMasters meeting with each other, bringing focus to 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 remember Ken Schwaber's discovery: “A dead ScrumMaster is a useless ScrumMaster.”)



Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

---

<sup>9</sup> <http://www.martinfowler.com/articles/continuousIntegration.html>

<sup>10</sup> <http://www.refactoring.com>

- 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 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?

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.

--mj  
Software Process Mentor  
Danube Technologies, Inc.

**danube**

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801