Scrum Tutorial

These notes provide additional thoughts to the video "Scrum Master in Under 10 Minutes" (http://www.youtube.com/watch?v=Q5k7a9YEoUI), in the order that the topics are presented in the video.

Product Backlog

- He does a good job of describing this
- It **is** a wishlist
- It **is** collaborative. Executives, developers, testers, etc. can all contribute to it.
    - Either formally (writing items) or informally (influencing the product owner to write items)

Roles

- He also does a good job of describing the product owner
    - Product owner **is** the filter of all the ideas that are out there
    - Product owner **does** set direction
- In my experience, the Scrum master should do little else than coordinate efforts and make sure the software is high-quality and progressing well
- Product Owner: Jay?
- Scrum Master: John?
    - I have Scrum experience
    - I won't be doing development, so I can dedicate all of my resources to making sure the project is progressing
- Subteams are sometimes used, but they probably won't be necessary for this project. We have a pretty small development group.

Release Backlog

- Rarely mentioned in literature that I've seen
- One of the top names in Scrum makes a convincing argument against it: http://blog.mountaingoatsoftware.com/why-there-should-not-be-a-release-backlog
    - Basically, it's an unnecessary additional layer
    - Nothing can be promised that far in advance
    - Things change!
    - Instead, he suggests looking at things from another perspective
        - Just keep doing what you're doing. Work as normally.
        - Use Scrum data (Burndown chart) to estimate the team's "velocity" / speed of development
        - Use this velocity to estimate what can be done by a certain time
        - See: http://blog.mountaingoatsoftware.com/wp-content/uploads/predictingwherewefinish.jpg
- Whenever you hear the term "Release Backlog" in this video, mentally substitute "Product Backlog"

Estimating

- Yes, subject experts are very useful
- If we have trouble with estimating, we can try a simple, useful technique to help (planning poker)
- Better to do a few things well than a lot of things badly
- Be realistic. No one can work until 4am.
    - The story of the Chicken and the Pig
        - Chicken = Managers, Pigs = Developers
        - Chicken: "Let's open a restaurant named 'Bacon and Eggs'"
        - Pig: "You would be involved, but I would be committed."
            - Silly, but it makes an important point. Part of the purpose of Scrum is protecting the developers.
            - To some degree, developers need a tool that allows them to say "No." to a 2am, day-before-the-deadline feature request.

Sprints

- Length for this project: 2 weeks
    - That's what we're naturally doing anyway
- He is correct that a late Sprint suggests a late product
- Every Sprint should create a "ship-ready" product, a "complete slice"
    - Something that the team can fall back on. "Funding dried up, but at least the last release does XYZ and does it completely."
    - The product created by a Sprint will not do everything, but it should provide a complete user experience
        - E.g., User can post videos, but cannot post photos. User can create an account, log in, and click "Post a video" to post a video. No link for "Post a photo" is provided, because the photo upload feature is not done yet.
        - It would make no sense to build video uploads before building user login. That would be an incomplete user experience.

Burndown chart

- Useful, but without the right tools it requires too much overhead. More on that in a moment.

Defect Backlog

- Rarely or never came across this in literature
- I suspect this is non-canonical like the Release Backlog
- In the past, I've mixed bugs with features in the Product Backlog
- For now, let's do that. If we need to track bugs separately (e.g., if the community reports many bugs), we can reconsider this

Daily Standup Meeting

- More important than he argues, especially for this group
    - Constant feedback loop. "Jimmy isn't finishing enough work, but Jimmy is always confused by the system architecture."
- Email / Wiki doesn't work well. Too much overhead.
- IRC may work. Development Seed does this successfully.
- Video chat might work well, but the (inevitable) technical issues might cause too much overhead.
- The "Standup" in "Standup Meeting" is actually very important.
    - Physical constraint to limit length of meeting
    - "My back hurts. Let's not drag this thing on."

Tools

- Trac is ok for Scrum, but not great
- Open Atrium is ok, but about on par with Trac. It's better in some ways, but worse in others.
- Wiki may not be enough (No list of "my tasks", no time tracking, no Burndown chart, etc.)
- I'll spend some time this week looking at other tools

Meetings

- He didn't talk much about these
- Planning meetings are essential
    - Small "breakout groups" are often necessary
        - Not everything can be decided in one, big meeting
    - In this meeting, the result of the Sprint should be carefully **designed**
        - Not just "a thing that does voting" and "a thing that does log-in" and "a thing that does photo upload" but "A system that allows the user to log in, post photos, view content, and vote on the content that exists."

    - Note that in this example, the detailed product design required us to realize that viewing content is important, which was not an area covered by the individual tasks.
- Sprint Retrospectives are very important, but often overlooked. An open, honest feedback loop is more important than it seems.

Definition of Done

- Spring 2010 in the OPL: "OCD Definition"
    - "Every task that we sign up for must be done perfectly. If we take on login in Sprint 1, we shouldn't have to touch login again until the software is due."
    - Not sustainable
    - Things change
    - If 12:00am Monday is the last chance to write the login instructions, you will stay up all weekend getting them just right
    - You will burn out. You will go crazy. Don't do it.
- Spring 2011 in the OPL: "Hippy Definition"
    - "The deadline is in your mind, man."
    - "Do what you can."
    - "If you can't get your work done in time, we'll move it to later."
    - Some Sprints contained half-done features
- This time: Something between the two. Take the benefits of each. Maybe...
    - Features are done (and well-done)
    - Features are polished
    - Features are tested
    - Features contribute to a complete user experience
    - No half-done work. If Sprints contain half-done work, we have nothing to fall back on.