1 Reading 4: HFSD Chapter 3 Pages 100-102, 104

The Big Board on your wall

- Once you know what you're building, you need to set up your software development dashboard for Iteration 1 of development.
- Your dashboard is a big board on the wall of your office that you can use to keep tabs on
  - what work is in the pipeline,
  - what's in progress, and
  - what's done.
- Burn down chart
  - Monitors how quickly you and your team are completing your work, measured in days on the vertical axis. This chart then plots how quickly you tick off your work remaining against the number of days left in iteration.
  - Work left: y-axis. Total work left in the **iteration** for all of your team.
    - Each unit (in the y-axis) is a day of work left on your user stories, starting at top and decreasing to 0 days at bottom.
  - Days left: x-axis. Days left in this iteration.
  - You have two lines on this graph
    - The ideal task burn-down rate
    - Your line, which represents your work against the days left. Plots above line means you're behind schedule, below line means ahead of schedule.

Be confident in your plans by applying velocity and not overworking yourself and your team. Remember that

- personal lives matter.
- fatigue affects productivity.

2 HFSD Chapter 4, Pages 109-123, 130-136, 139-142

Getting to the real work.

- Your work is more granular than your user stories.
  - Your user stories were for your users; they helped describe exactly what your software needed to do, from the customer's perspective.
  - Now that you need to start coding, you probably need to look at the stories differently.
    - Each story is really a collection of specific tasks, small bits of functionality that can combine to make up one single user story.
  - A **task** specifies a piece of development work that needs to be carried out by **one developer** in order to construct part of a user story.
  - Each task has a **title** so you can easily refer to it, a **rough description** that contians details about how the development of that task should be done, and an **estimate**.
  - Each task has its own estimate and the best way to come up with those estimates is by playing **planning poker** again with your team.
- Task estimates add confidence to user story estimates.
  - Your user story estimates kept you in the **right ballpark** when you were planning your iterations, but tasks really add another level of detail specific to the actual coding you'll do for a user story.
  - It's often best to **break out tasks from your user stories at the beginning of the estimation process** (maybe before planning iterations), if you have time.
  - Remember that your user story estimate was accurate enough in the beginning to let you organize your iterations. With task estimates, you have a set of more accurate data that either backs up your user story or conflicts with them.
  - Your task estimates should ideally be between $\frac{1}{2}$ and 5 days in length. A shorter task measured in hours is too small a task. A task that is longer than 5 days spreads across more than one working week – might lose focus.
  - If you discover that you missed a big task, you might need to readjust your iteration.
- Plot just the work you have left
  - Every time we do any work or review an estimate, we update our new estimates, and the time we have left, on our burn-down chart.
- Add your tasks to your board
  - Update the big board.
  - Add an in progress, and complete, section for tracking tasks and user stories. You can also add a Next section in case a user story has to get bumped from iteration.
  - Sticky notes are good for tasks – they can hang on the bottom of the user story they belong to.
  - Note that the origianl user story estimates are gone after you have the task estimates. You're relying on the combined estimates of all tasks for all user stories for iteration.
- Start working on your tasks.
  - Write down initials of developer working on each task on its sticky note.
  - No hard rule for assigning tasks to people. Just see who would be most productive or – if you have the time, will learn most from a particular task by looking at their own expertise – and then allocate the task to the best-suited developer, or the one who will gain the most, that's not already busy.

  - Prefer allocating all tasks from one story instead of one task for each story. That way, you don't end up with multiple stories in a half-done state.
    * If you have one story your other stories depend on, get that story's tasks done at once.
    * If your stories are independent of each other, you may work on tasks from multiple stories all at the same time.

- A task is only in progress when it's in progress.
  - Only put tasks that are actually being worked on in the In Progress column.
  - If it's assigned but not being worked on, then it's not in progress.
  - Your board is only as valuable as it is accurate.

- What if I'm working on two things at once?
  - Not all tasks are best executed in isolation. Sometimes, two tasks are related and, because there's os much overlap, it's actually more work to tackle one, and then the other separately.
  - Sometimes, working on both tasks at the same time is the best option!
  - Some rules to think about
    * Double-up on tasks that are related to each other, or at least focus on roughly the same area of your software.
    * Try not to double-up on tasks that have large estimates.

- Your first standup meeting
  - to keep everyone in the loop, while not taking up too much of their time, you conduct a quick standup meeting every day.
    * A meeting so quick you don't even have time to sit down (hence "standup").
  - Your daily standup meetings should
    * Track your progress
    * Update burn-down rate
    * Update tasks
    * Talk about what happened yesterday and what will happen today
    * Bring up any issues
    * Last between 5 and 15 minutes.
  - A daily standup meeting should keep everyone motivated, keep your board up-to-date, and highlight any problems early.

- End of the week meeting.
  - Ideally aim for 5 minutes in the standup meeting.
    * Main questions to ask: are there any issues? Have we finished anything?
  - If an issue comes up in standup meeting that will take some discussion to resolve, schedule another meeting specifically for that issue.
  - Standups should be held on a daily basis.
  - Standups should ideally be the first thing in the morning, but you may not be able to all meet in the mornings (e.g., remote employees).
    * Standups should be conducted when majority of team begin their working day.
    * You may, rarely, split standup meeting in two. If so, updating the board is critical!

- What if the customer calls with a last-minute request?
  - You have to track unplanned tasks.
  - Unplanned tasks are added to an extra user story. Use a red card for unplanned task so you can tell them apart from regularly planned tasks.
  - Talk to the customer about it. Remember that the customer sets priorities, not you.