Today

- Velocity and planning (more info)
- Taking stock (where we are and where we’re going)
- Presentation description and some tips

Homework

- HW6 due
- HW7 out
Overview of basic approach to planning:

1. Finalize initial backlog for MVP
2. Prioritize backlog items
3. Estimate backlog items (in story points)
4. **Estimate velocity (e.g., story points per sprint)**
5. **Revise scope as needed (based on estimates & velocity)**
6. **Develop milestones (way points) & sprint release plan**
How long will your project take?

First approach
- say you add up stories/tasks and get 364 dev days
- and you have 4 developers
- \(364 \div 4 = 91\) calendar days \(\approx 3\) months

Q: Is this a realistic estimate?
- think of the days on a calendar
- don’t want to sign up for working weekends!

Second approach
- say your target is in 90 calendar days
- there are only 60 dev days in 90 calendar days!

\[
\begin{array}{cccccccc}
\text{mo} & \text{tu} & \text{we} & \text{th} & \text{fr} & \text{sa} & \text{su} \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
10 & 11 & 12 & 13 & 14 & 15 & 16 \\
17 & 18 & 19 & 20 & 21 & 22 & 23 \\
24 & 25 & 26 & 27 & 28 & 29 & 30 \\
\end{array}
\]

- \(364\) dev days \(-\) \((4\) devs \(\times\) 60 dev days\) \(=\) 124 dev days
- need an additional 124 dev days!

Q: What should we do?
- reduce scope (stories), and/or
- increase cost (devs), and/or
- increase time (commitment)
Q: we still have a problem ... what is it?

- we are assuming everyone is 100% productive every day
- vacation, illness, software/hardware updates, paperwork, ...

**Third approach**

- We need to account for our team’s “velocity”
- measure of how “productive” team is
- alternatively, how off our estimates tend to be
- relies on past estimates and actuals

**Velocity**

- measured as % of productive work
  \[
  \text{Velocity} = \frac{\text{EstimatedDaysOfWork}}{\text{ActualDaysRequired}}
  \]

- Q: How many days needed to do \( n \) estimated days of work?
  \[
  \text{DaysRequired} = \frac{\text{EstimatedDaysOfWork}}{\text{Velocity}}
  \]

- have to keep track of totals through lifetime of (and across) projects
- notice the number of days keep shrinking!
  - 30 calendar days = 20 days of work
  - 20 work days = 14 productive days (if 70% velocity)
Velocity ... An alternative definition ... see textbook

The average number of story points per sprint

• given a backlog of $n$ story points,
• team velocity $v$ (in story points / sprint),
• and 2 week sprints
• need $n/v$ sprints ... or $2 \times (n/v)$ total weeks for 2-week sprints

Exercise: What do you think your team’s velocity is?

• Consider both “definitions” of velocity ...
• How accurate have your estimates been?
• How many story points per sprint (on average)?
Scheduling: Release Planning

Scheduling: mapping what you need to accomplish in your project to a timeline and then monitoring your progress to ensure you deliver your product on time

- Our goal is to come up with an initial/rough “schedule”
- Which we are calling the sprint release plan
- This is in contrast to a detailed schedule (Gantt Chart)
  - Word Breakdown Structure (WBS) + Dependencies + Schedule

Example Gantt Chart
Milestones

A "milestone" is an important project event or accomplishment

- e.g., when a feature area is completed
- others: components integrated, alpha/beta release, deployment, testing begins/ends, data obtained, etc.

Milestones are an important part of scheduling

- determining when milestones need to occur (planning)
- “checkpoints” to see if project is on track (monitoring)

Defining project milestones

What are the major accomplishments & events along the way?

- consider milestones for all the project deliverables
- consider major feature areas
- consider both technical and non-technical areas
- consider dependencies
- consider testing
- consider deployment (e.g., app submission)
Defining project milestones

Example milestones for your projects ...

- final version of UI design for XYZ features
- initial version of DB schema
- version 1, 2, 3, ... of XYZ features
- final version of authenticated login
- first submission to app store, final submission
- migration to deployment server
- initial set of usability tests complete for v1 features
- system tests for XYZ features
- web service API feature freeze

Exercise: Come up with a list of milestones for your project

- Consider all deliverables (and dependencies) ...
- E.g., documentation, deployment, system testing, usability testing, deployment
Using milestones in scheduling

Goal is to come up with a schedule for your project ...

- we will be using “two-week sprints/iterations”
- after project plan, there are \( \approx 22 \) weeks \((7 + 15)\)
- each sprint will be a “mini” milestone (visible features)
- these should lead to the “major” milestones

What are the goals for each sprint?

- **First**: when do the milestones need to be completed?
- **Then**: what will you do each sprint to ensure this?
- **While**: keeping users engaged ... focus on visible progress
Taking Stock ...

Big Picture

- Week 7 (this week): Finalize your project plan
- Week 8 (next week): Presentation & incorporate feedback into plan
- Week 9: Start development!
- Weeks 9–14: Working toward end-of-semester demo/prototype

<table>
<thead>
<tr>
<th>Weeks</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
<th>S7</th>
<th>“S8”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–2</td>
<td>3–4</td>
<td>5–6</td>
<td>7–8</td>
<td>9–10</td>
<td>11–12</td>
<td>13–14</td>
<td>15–16</td>
<td></td>
</tr>
</tbody>
</table>

* We are here

Your end-of-semester demo/prototype must:

Consist of a substantial/meaningful portion of your system
  - “meaningful” means meaningful to users/sponsors

Be created following good design principles
  - you will need to provide evidence of this

Be well tested
  - also will need to provide evidence of this
Project Plan Presentation

Overview

- Aim for 12-15 minutes
- Every team member must speak roughly equal amount of time
- Plan for a range of questions (from DAB, Sponsors, Faculty, ...)
- Practice, practice, practice ... and then practice!
- Must do a practice presentation with faculty advisor ...
  - Need to arrange this
  - Don’t waste their time!

Presentation Audience

- Mix of people that know and don’t know about your project
- In general, assume most don’t know anything about your project or work
- Focus on demonstrating you understand needs and have a good plan!
General Outline (* this is a suggested order/breakdown)

1. Project overview (∼2 min) ... the “why” and “what”, briefly
2. Deliverables and scope (∼2 min) ... deliverables & architecture
3. Requirements and design (∼4 min) ... major features, UI/functionality
4. Major work accomplished (0 – 2 min) ... if work already done
5. Milestones and product release plan (∼3 min) ... make plan clear & justify
6. Maintenance considerations (∼1 min) ... the main/major issues
7. Project risks and strategies (∼2 min) ... risk, plan b, monitor, triggers
8. Project management considerations (∼1 min) ... work process & tools
Presentation Tips

(1) Keep slides simple, clean, concise, and organized
   - professional doesn’t mean fancy transitions or templates

(2) Slides should be easy to read and follow
   - Don’t use small fonts (should be able to read easily from back of room)
   - Keep text to a minimum
     - don’t need complete sentences
     - keep bullet text to one line
   - Don’t over bulletize (do you really need sub-bullets?)
   - Focus only on the main points ... say the rest as needed
   - Slides should be cues

(3) Use figures to express ideas over text
   - Diagrams, UI sketches, timelines, workflow/dataflow, architecture
   - For every slide, consider if a figure makes sense ...
   - Consider designing slides around figures (figure firsts, then text)
   - Try “storyboarding” your presentation (divide paper into 4 sections ...)

(4) Talk to the audience
   - Look at the audience when talking
   - Don’t look at the computer or screen
   - Don’t point at the computer, but can point at screen (if needed)
(5) Don't rush through or use “fast-talk”
- Know the main points you want to say and stick to them
- Don’t try to cram information
- Don’t rush through main points (fast talk)
- Highlight main points (e.g., small pauses)

(6) Practice, practice, practice ... then more practice!

(i). Practice Individually:
- Figure out what the main points you want to say
- Practice your parts on your own
- Keep track of your time as you practice
- Ditch the helper cards ... they are unprofessional
- Get rid of “umm”, “so”, and other filler phrases

(ii). Practice as a Team:
- Content, timing, slide quality

(iii). Practice with your Advisor:
- Best feedback if you are giving your close-to-final presentation
- Ask for potential questions

(iv). Repeat individual & team practice for final presentation