Exercise: Redo your user story estimates (use planning poker)

1. Make a “best case” estimate (min possible time/points)
2. Make a “worst case” estimate (max possible time/points)
3. Make a “most likely case” estimate (under “normal” conditions)
4. Your “expected case” estimate (avg over long run … 50% conf.)

\[
\text{Expected} = \frac{\text{Best} + (4 \times \text{MostLikely}) + \text{Worst}}{6}
\]

... from Program Evaluation & Review Technique (PERT)
5. To find 90% confidence, find standard deviation

\[ \text{StdDev} = \sqrt{\left(\frac{\sum (\text{Expected} - \text{ExpectedMean})^2}{n}\right)} \]

- if \( n < 10 \), can approximate as \( \frac{1}{6} \) range:

\[ \text{StdDev} = \left(\frac{\sum \text{Worst} - \sum \text{Best}}{6}\right) \]

6. 90% confidence:

\[ \text{Expected} + (1.28 \times \text{StdDev}) \]

- watch for precision (e.g., 120 days vs 118.64)

* assumes estimates follow a normal distribution

---

**Estimation (cont)**

**How well does this really work?**

- Estimation only improves from experience
- Being aware of best, worst, most likely can help …

**Caution:**
- Most developers “worst case” more like “best case”
- We tend to underestimate!!!

“You never have to fear that estimates created by developers will be too pessimistic, because developers will always generate a too-optimistic schedule.”

- Chris Peters, Microsoft
Better Estimates Through Tasks

Easier to estimate “piecewise”

Break each story into a set of tasks

- a task is a **unit of work** to get a story implemented
- a task alone is a non-feature
  - stories are features (user cares about)
  - tasks are what developers work on (user rarely cares)
- a task is typically done by one developer (**owner**)

Better Estimates Through Tasks

Stories fit on 3x5 cards, tasks fit on post-it notes

- title + terse description
- an estimate (via p.p.)
- note: design and test work are also tasks!

Task estimates usually more accurate

- sum task estimates to get better story estimates
Exercise

Give it a try …

- break your stories into tasks
- estimate time for each task (e.g., worst or most likely)
- sum for task estimate to get story time estimate
- compare the results

Estimation & Planning

What you need to do:

1. Finalize initial backlog for MVP
2. Prioritize backlog items
3. Estimate backlog items (in story points)
4. Estimate velocity (story points per sprint)
5. Revise scope as needed (based on estimates & velocity)
6. Develop milestones (way points) & sprint release plan
Estimation & Planning

What you need to do:

1. Finalize initial backlog for MVP
2. Prioritize backlog items
3. Estimate backlog items (in story points)
4. Estimate velocity (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

- let's say you add up stories/tasks and get 364 dev days
- and you have 4 developers

\[
364 / 4 = 91 \text{ calendar days} = 3 \text{ months}
\]

Q: Is this a realistic estimate?

- think of the days on a calendar
- don’t want to sign up for working weekends!
How long will your project take?

Second approach

- let's say your target is in **90 calendar days**
- there are only **60 dev days** in 90 calendar days
  
  \[364 \text{ dev days} - (4 \text{ devs} \times 60 \text{ dev days}) = 124 \text{ dev days}\]

- we're **over by 124 dev days** (>> 20%)

Q: **What should we do?**

- reduce scope (stories), and/or
- increase cost (devs), and/or
- increase time (commitment)

---

How long will your project take?

Third approach

Q: we still have a problem … what is it?

- we're assuming everyone is 100% productive every day
  
  - vacation, illness, software/hardware updates, paperwork, …

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
How long will your project take?

**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 project
  - or even 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**

The average number of story points per sprint

- Given a backlog of \( m \) story points,
- The team velocity \( v \) (in story points / sprint),
- And 2 week sprints
- Need \( \frac{m}{v} \) sprints (or \( \frac{m}{v} \times 2 \) total weeks if 2-week sprints)

Can convert to a range with min and max velocity, etc.
Exercise …

- What do you think your team’s velocity is?
  
  Do both “definitions” of velocity …
  
  - How many story points per sprint?
  - How accurate have your estimates been?

Estimation & Planning

What you need to do:

1. Finalize initial backlog for MVP
2. Prioritize backlog items
3. Estimate backlog items (in story points)
4. Estimate velocity (story points per sprint)
5. Revise scope as needed (based on estimates & velocity)
6. Develop milestones (way points) & sprint release plan
Release Planning (Scheduling)

**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)

Example Gantt Chart
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

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

- Consider all deliverables (and dependencies) …
- E.g., documentation, deployment, system testing, usability testing, testing “in the wild”, ...
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 ~ 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

Using milestones in scheduling

When scheduling your milestones … think about
  ○ epic/story priorities (suggest order of development)
  ○ areas likely to change
  ○ developing your project in iterations (prototypes)
  ○ visibility (what users/customers see & care about)

Note about Agile/Scrum ...
  ○ schedules can be hard to define when lots of uncertainty
  ○ detailed/complex schedules often go awry
  ○ focusing on milestones & sprints is still helpful
**Exercise**

**Exercise:** Determine a rough plan for when the milestones should be completed

- Again, consider all deliverables …
- When should you be “done” with (v1) your product?

---

**Initial Sprint Release Plan**

**What is the goal of each two week sprint?**

- What stories will be implemented?
- What features will be made visible to the client?
- Are we making progress towards major milestones?
Initial Sprint Release Plan

For your project plan …

<table>
<thead>
<tr>
<th>Sprint Date</th>
<th>Spring Goal</th>
<th>Backlog</th>
<th>What we will demo</th>
</tr>
</thead>
<tbody>
<tr>
<td>4th Week in Oct to 1st week in Nov</td>
<td>Brief summary of the overall goal of the sprint</td>
<td>Requirements from Table 2 that will be finished</td>
<td>What aspects will you show users/sponsor to get feedback on</td>
</tr>
<tr>
<td>2nd Week in Nov to 3rd Week in Nov</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>4th Week in Nov to 1st Week in Dec</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Risks …

“Post mortems” vs “Pre mortems”

Exercise …
What are potential project risks? …

- New technology (e.g., PLs and frameworks)
- Dependencies (other teams, other systems)
- Inexperience (specific technology or techniques)
- Unchartered territory (novel technical approaches)
- Unclear/changing requirements
- Communication difficulties
- Lack of buy in or shaky benefits
- ...

Risks -- Identify, Monitor, Mitigate

Identify and make a plan for your project risks

1. Write down (describe) the risk
2. Explain why it is a risk and its potential impact
3. What you will do to initially prevent it
4. What events/situations will trigger mitigation (a Plan B)
5. What you will do if risk becomes reality (what is Plan B)

* This is what is needed for your project plan (identify, monitor, mitigate)
Exercise: In your team, identify a project risk:

- How will you try to prevent it?
- How will you monitor it?
- What is your Plan B?
- What events/situations will trigger change in plan?

Maintenance Plan ...
Wrapping up the project plan ...

**Maintenance Considerations (Sect. 6)**

1. Once project is completed, **who** will maintain it?

2. **What** will maintenance entail?
   - Bug fixes?
   - Modifications/extensions?
   - Further integration into production system?

3. **How** will you develop/design the system to help?
   - Are there design decisions to make maintenance easier?
   - Documentation?

The rest …
Wrapping up the project plan ...

Project Management Considerations (Sect. 7)

How are you organizing yourselves as a team?

- When/where are your weekly team meetings
- When/where are you meeting your sponsor, etc.
- What tools are you using?
- How are you breaking up the work?
- Who is responsible for what?
- ...

Wrapping up the project plan ...

The Appendix

You can use for any additional info ...

- E.g., additional diagrams (from your sponsor)