Planning for Success (aka “Avoiding Failure”)

OUR PROJECT PLAN IS SO COMPLICATED THAT FAILURE IS ASSURED.

BUT COMPLEXITY IS TOO ABSTRACT FOR YOU TO MANAGE, SO INSTEAD YOU WILL SPRAY MY ENERGY INTO THE VORTEX OF FAILURE.

GO.

I NEED YOU TO FINISH IT SIX WEEKS SOONER FOR A TRADE SHOW.

AFTER EIGHT MONTHS, SENIOR MANAGEMENT FINALLY APPROVED YOUR PROJECT PLAN.

IT’S TOO LATE. ALL OF THE TECHNOLOGY HAS CHANGED AND OUR COMPETITORS HAVE LEAPFROGGED US.

MAYBE YOU COULD WRITE A NEW PLAN.

OR WE COULD GET THE SAME RESULT BY RESUBMITTING THIS ONE.
Project Planning

You have to have a plan to know that you’ve fallen off the plan.
Goals of Project Planning

- Plan to deliver working product
  - on budget
  - on time

1. Staffing of project to move at required speed

2. Identify first release of working product
   - Prioritize what customer/market needs
   - As soon as possible

3. Identify additional releases through project completion
Overriding Principles of Planning

- Iterative product delivery
  - Establish intermediate product milestones
- Be realistic
  - Don’t set up customer for a surprise
- Be quantitative
  - Measure, measure, measure
- Make it visible
  - Seeing is knowing
New Concept: **Milestone**

Each iteration is **QUALITY** software.

Milestone 1.0
- Baseline
- Minimal Subset
- Minimal Viable Product

Milestone 2.0
- Working product (not a demo)
- Additional work to release
- More feedback (use/market)
- Get paid (or graded :)

... 

Milestone N
- (Final Product)

Earliest possible date
New Concept: **Baseline**

“Without the ability to load, edit, & save a document with text... a word processor is not useful.”

- With the customer, identify the Baseline
- Part technical: need login to make a purchase
- Part is getting over “nice to have”
  - Don’t need reviews for airline reservation System
- Some of this is being very clever
  - Break up user stories into smaller, simpler pieces
  - Reviews of flights are just stars or text, but not both
  - Skip identities of reviewers, top 100 reviewers, etc.
  - Outsourcing reviews of flights (i.e., to Yelp or Expedia)
  - Can deliver more later!!!
Baseline for myCity?

- Display birds-eye map with user at the center (H)
- Show GoogleTalk friends on the map (H)
- Click friend, get a textbox, type & send msg (H)
- Map continues to track user’s changing location, with option to not track (M)
- Map updates with coming & going of friends (M)
All part of **making a large project act like a small one**

Different frequencies for different types of “ship” manages the **tradeoff** between **cost** of shipping and **value** of feedback.
New Concept: **Velocity**

- Can you get 30 days of work done in month?
  - No! Need down-time on weekends to “refuel”

- Can you get 8 hours of programming in a day?
  - No! Meetings, vacation days, ........................................

- Long-term solution is **not** overtime
  - Costly mistakes, morale problems, burnout

- Two key adjustments
  - Count 20 days to a month
  - **VELOCITY** to account for work-day inefficiencies
New Concept: Velocity

Velocity is a trendy word for efficiency

Due to *likely*, recurring reasons to slow work (called *overhead*)

Range from 0 to 1 (i.e., 0% to 100%)

\[
\text{days of work} \div \text{velocity} = \text{work days required to get work done}
\]

- If “show map around user” user story is estimated at 3 days
- And velocity is 0.7
- Then work days to complete = \(3 / 0.7 = 4.3\)
New Concept: *Velocity*

What’s your team’s net productivity per month?

\[
\frac{\text{days of work}}{\text{velocity}} = \text{work days required to get work done} \times \text{velocity}
\]

\[3 \times 20 \times 0.7 = 42\]

- The number of people on your team.
- 20 working days in your iteration.
- Your team’s first pass velocity.
- The amount of work, in person-days, that your team can handle in one iteration.
How get your team’s velocity?
Guess, then measure

- Initially guess velocity
  - Past projects, rough estimate, “back of the envelope”

- But then, measure
  - Use previous user stories, iterations, milestones

- “The past is prologue”
  - Means: the past predicts the future
  - Don’t think, “We’ll get the hang of it and improve”
  - Good engineer’s are pessimistic

- Customer’s hate negative surprises
  - Under promise and over deliver
More people sometimes means diminishing returns \(\text{(lower velocity)}\)

Adding more people to your team doesn’t always work as you’d expect. If 1 person takes 273 days to complete Milestone 1.0, then 3 people **won’t** necessarily take 91. In fact they could actually take much longer! Take a look...performance doesn’t always increase with the size of your team:

\[
\begin{align*}
\text{Performance} & \\
\text{...or rate of development work actually being done} & \\
\text{About here, your team’s performance starts to max out.} & \\
\text{As your team gets even bigger all the extra communication paths mean that adding new members to the team has less and less impact on performance.} & \\
\text{For small teams, and when startup and setup time is factored in, you can see a big improvement when adding extra people.} & \\
\text{Beyond a certain point, adding extra people can actually reduce the performance of your team.} & \\
\end{align*}
\]

Often, Less is More
Make it Visible: The Big Board

User stories

Progress of user story implementation...

Can see: what’s getting worked on, progress, completed, todo
Make it Visible: The Burn Down Chart

**Burn Down:** ahead/behind schedule

- **Days of Work left**
- **Work Days left**

The total work left in the iteration for all of your team. Each unit is a day of work left on your user stories, starting at the total days at the top and decreasing to 0 days at the bottom.

The ideal task burn-down rate:

You’ll plot your work against the days left. Plots above the line mean you’re a bit behind schedule. If you’re plotting below the line, you’re ahead of schedule.

Days left in this iteration
Take-Aways

Development Techniques

Iterations should ideally be no longer than a month. That means you have 20 working calendar days per iteration.

Applying velocity to your plan lets you feel more confident in your ability to keep your development promises to your customer.

Use (literally) a big board on your wall to plan and monitor your current iteration’s work.

Get your customer’s buy-in when choosing what user stories can be completed for Milestone 1.0, and when choosing what iteration a user story will be built in.

Development Principles

Keep iterations short and manageable.

Ultimately, the customer decides what is in and what is out for Milestone 1.0.

Promise, and deliver.

ALWAYS be honest with the customer.

BULLET POINTS

- Your customer prioritizes what is in and what is out for Milestone 1.0.
- Build short iterations of about 1 calendar month, 20 calendar days of work.
- Throughout an iteration your software should be buildable and runnable.
- Apply your team’s velocity to your estimates to figure out exactly how much work you can realistically manage in your first iteration.
- Keep your customers happy by coming up with a Milestone 1.0 that you can achieve so that you can be confident of delivering and getting paid. Then if you deliver more, they’ll be even happier.
Milestones and Risks

What risks do baselining and milestoneing address that user stories and iterations don’t?

A. Things relating to a real, shipped product:
   - **Risks of late shipping**
     - beat to market
     - not getting paid
   - **Risks of late feedback; learn about:**
     - real users in real use
     - Scale
     - OS/hardware version compatibility
       - maybe performance, too
     - Marketing issues (competition, features, pricing)
     - Product support
     - I bet you can think of some more

B.

C.

D.
Beating Scale

Name a technique or principle from today/Ch 3 for making a big project act like a small one

A. Be quantitative: measure

B. Milestones

C. Hire enough programmers to get high velocity

D. See progress at a glance on the Big Board

E. See if on schedule with the Burn Down Chart

- A, D, and E are important, but don’t make a large project act like a small one
- Velocity is about team efficiency, not speed. Adding programmers may increase speed, but will typically decrease efficiency.
Take-Aways from Class Today

- There’s no free lunch
  - More, Sooner, Cheaper
  - Pick any two: Iron Triangle

- Sooner is the way to go
  - Gets you feedback from market
  - Insight on other risks
  - Revenue (good for customer and pays next milestone)

- Milestoning, Baselining
  - **Sooner** (with less functionality)
  - But insight for better **functionality**, better **cost** estimation
  - Promise and Deliver – no late surprises!
    - Not just about happy customer
    - You get paid, you get to sleep, you have good rep.

- Velocity
  - Reigning in optimism
  - Promise and Deliver – no surprises!