PROJECT MANAGEMENT

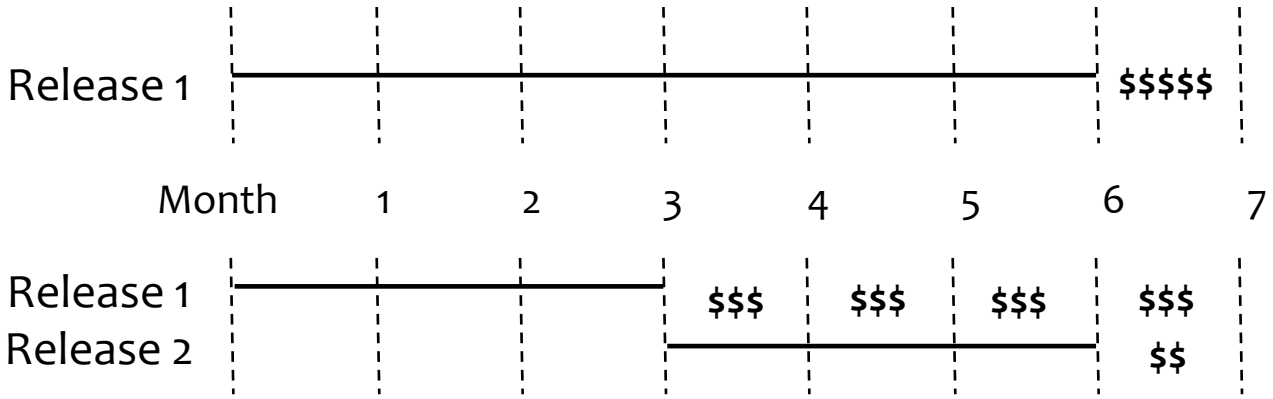
Part 3 : Planning

Contents : Releasing, planning, slacks and estimate

Tianxiao LIU - Master IISC 1 – University of Cergy-Pontoise

RELEASE EARLY AND RELEASE OFTEN

- Group the most valuable features together and release them first
  - To achieve startling improvements in **value**



FREQUENT RELEASES : BENEFITS FOR DEVELOPERS

- Releases are **painful**, aren't they ?
  - Flag days on the schedule
  - Repository freezes
  - Rushes to complete everything…

- Delivering tested, working and valuable software regularly

  → Increase trust between stakeholders and you

  → Get feedback very quickly

  → Learn and adapt fast

- If we can release at any time, that'll eliminate all the stress.

HOW TO RELEASE FREQUENTLY

Releasing frequently ≠ Setting aggressive deadlines

- We need to recognize a Minimum Marketable Feature (MMF)
- MMFs may provide value in many ways
  - Competitive differentiation
  - Revenue generation
  - Cost savings

- Group MMFs into different releases
  - Team brainstorming exercise
  - Challenge : **how to make small releases ?**

KEEP YOUR OPTIONS OPEN WHEN PLANNING

- Build a plan that allows you to release at any time.
  - Not to **actually** release all the time → **enable** you to release at any time.
- What benefit do we have by doing this ?
  - An important and new opportunity comes → immediately change directions to take advantage of the opportunity.
  - There is some sort disaster (ex. project's surprise cancellation) → We can release what we have.
- Financial issue : **At any time, you should be able to release a product that has value proportional to the investment you've made.**
- Technique : build the plan so that each task stands alone.
  - Use vertical stripes instead of horizontal stripes
  - Ex. (get data, validate data, write data to DB) **VS** (process customer data, process shipping address, process billing information)

WE NEED SLACK IN OUR PLAN

- Our project plans can't be disrupted by the slightest provocation.
- The amount of slack depends on the randomness of problem.
- Introduce slack
  - Schedule no work on the last half-day of your plan ?

    → This gives us the slack, but it would be pretty wasteful…
- A better solution
  - Schedule useful and important work that isn't time-critical.
  - This kind of work can be set aside in case of an emergency.
  - Ex. **Paying down technical debt**

SLACK : RESEARCH TIME

- Programmers must continually improve their skills.
  - Keep up with their constantly expanding field
  - Learn things that enhance their work on the project
- Solution
  - Set aside half a day for each programmer to conduct self-directed research on a topic of his choice.
  - During this time, no modification on the project code source.
- Recommendation
  - A quick stand-up meeting to ask that people share what they've done in informal peer discussion. → share knowledge

ESTIMATING AND VELOCITY

- One of the most difficult things programmers must do…
- They find that they consistently estimate too low.
  - Magical approach : multiplying by three ?!
- It's not easy to predict how we spend our time
  - Interrupted concentration and surprising emergency.
- Estimates are never accurate, but they are **consistently** inaccurate.
- **Team or individual velocity**
  - The number of (function points or story points) that team can accomplish at an iteration (day, week, etc.)
  - Instable velocity at the beginning → stabilized after several iterations.

EXPLAINING ESTIMATES

- **One thing is always true** : customers and stakeholders are invariably disappointed by the estimates.
- Comments of disappointment should be treated as straightforward requests for information.
- "Why does that cost so much ? "
- List the issued you considered when coming up with the estimate.
- Your initial, gut-feel estimate is most likely correct.
- Only change your estimate if you learn something genuinely new.
- Never change it just because you feel pressured → professionalism.

AFTER THE PLANNING SESSION

- **After we finish planning the releases, work begins !**
- So how do we deliver on our commitment ?
- Programmers volunteer to work tasks
  - The may ask for pairing.
  - Pairs break apart as they finish their task.
  - Individuals pick up new tasks from the board and form new pairs…
- As work continues, we need to revise the plan to reflect the changing situation.
  - Keep track of original task estimates (for better estimate later)
  - Small demonstration for new value added into the product