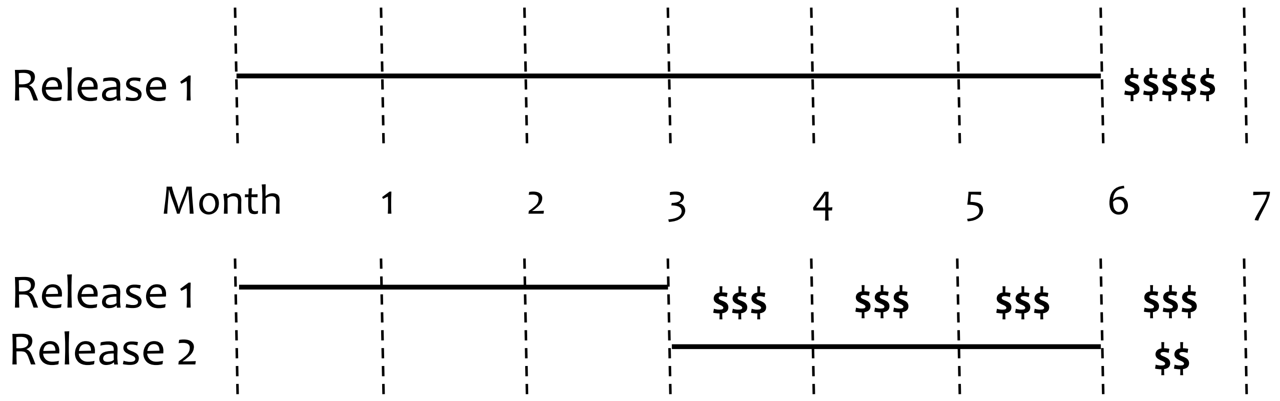
PROJECT MANAGEMENT

Part 3 : Planning and agile practice

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. (process customer data, process shipping address, process billing information) **VS** (get data, validate data, write data to DB)

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

AGILE PROJECT VALUES

- **Values are abstracts, but also identifiable and distinct**
- **Courage**
  - To make the right decisions, even when they are difficult
- **Communication**
  - To give the right people the right information : maximum use
- **Simplicity**
  - To discard the things we want but don't actually need
- **Feedback**
  - To learn the appropriate lessons et every possible opportunity
- **Respect**
  - To treat ourselves and others with dignity
  - To acknowledge expertise and our mutual desire for success

UNDERSTAND DEEPLY YOUR PROJECT

- **Improve your process by understanding how it affects your project**

- **Take advantage of feedbacks**
  - From everybody and everything : program, team, customers, supervisors…
  - What works well and what doesn’t
  - Pay attention to what’s happening around you
  - Ask very often : why are we doing this practice ?
  - A complaint is interesting if it is based on an element of truth

TUNE AND ADAPT IF NECESSARY

- **When you see the need of a change, modify your process.**
- **Your project team is unique !**
  - For every team, the needs are different.
- How to tune : in an agile way too
  - Experiment it carefully : make small, isolated changes that allow you to understand the results.
  - Be specific about your expectations and about the measurements for judging success.
  - Use the results of your experiments to make further changes and iterate until you are satisfied with the results
  - Please have the courage to experiment and occasionally fail.

BREAK THE RULES WHEN YOU SHOULD

- **Rules are important because they exist for a reason.**
- **However, rules can not anticipate all possible situations…**
- **Establish rules for your team**
  - Find the reason for each rule
  - Exercise pragmatic idealism : establish an underlying set of ideals based on practical results.
- Be prepared to explain your experiment
  - It’s easier to be understood when you are breaking rules and you can demonstrate that your are trustworthy and effective

RELY ON PEOPLE : BUILD EFFECTIVE RELATIONSHIPS

- **You are not working alone**
  - You need deal with other person during the process
  - **A grudging detente* is not enough !**
  - You need to form solid working relationships : honesty, trust, cooperation, openness and mutual respect
- Forcing does not work
  - Have people sit together and collaborate in pursuit of common goals
- Blame-oriented cultures sabotage relationships
  - Credit and being right are not important.
  - **Treating others with respect and cooperating to produce great results is important.**

* FR : Détente à à contrecœur

AN EXAMPLE FOR RELATIONSHIP BUILDING

- **One team member X has an abrupt communication style**
  - This leads to friction with people who don’t know him well
  - X is being rude ??
- The truth
  - X is just not a native language speaker and is laconic by nature !
- Lesson
  - It’s always easy to assume the worst about someone’s motivation when you can’t talk face-to-face
- Solution : meet this person as often as possible
  - You will find that nothing is personal, but rather an artifact to his culture.

LET THE RIGHT PEOPLE DO THE RIGHT THINGS

- **We need to diverse range of expertise.**
- **Within the project team, anyone can be a leader**
  - Encourage team members to turn to the person or persons most qualified to make a necessary decision
- Leadership means ?
  - "I'm the most senior, so we will do it in my way !" → NO
  - Don't act as if you have authority over them if you are not the most qualified to make a decision.
- Real managers
  - They manage but they don't do all by themselves.
  - Let team members tell you what they need you to do to help them succeed

ELIMINATE WASTE

- Work in small reversible steps
- **Reduce the amount of work you may have to throw away**
  - Breaking your work down into its smallest possible units and verify them separately
  - Do not solve multiple problems together
  - Make incremental change → better approach
- Maximize work not done : “don’t eat too much at one time”
  - “Simplicity is the art of maximizing the work not done.”
  - Resist to solve big, hairy problems
  - For eliminating waste and make your process more agile : do less at each step

SEEK TECHNICAL EXCELLENCE

- **What is the intellectual basis of a system design ?**
- **What does it mean to have a good design ?**
  - **Problem : many discussions of good design focus on specific techniques**
  - **Good is not obvious.**
- Quality without a name (QWAN)
  - An ineffable sense of rightness in the design
  - This is too vague !
- Design for understanding
  - Example : design an airplane : trade off safety, flue efficiency, passenger capacity and production cost.
  - A good system design minimizing the time required to create, modify and maintain the system while achieving acceptable runtime performance