Project Management (Part 1)

Week 3

"Management is about persuading people to do things they do not want to do, while leadership is about inspiring people to do things they never thought they could."

– Steve Jobs

Human-Centered Computer Systems Lab

HCSLAB

SEOUL NATIONAL UNIVERSITY

Objective

- To understand the challenges in project management
- To learn how to manage a software project

List of Contents

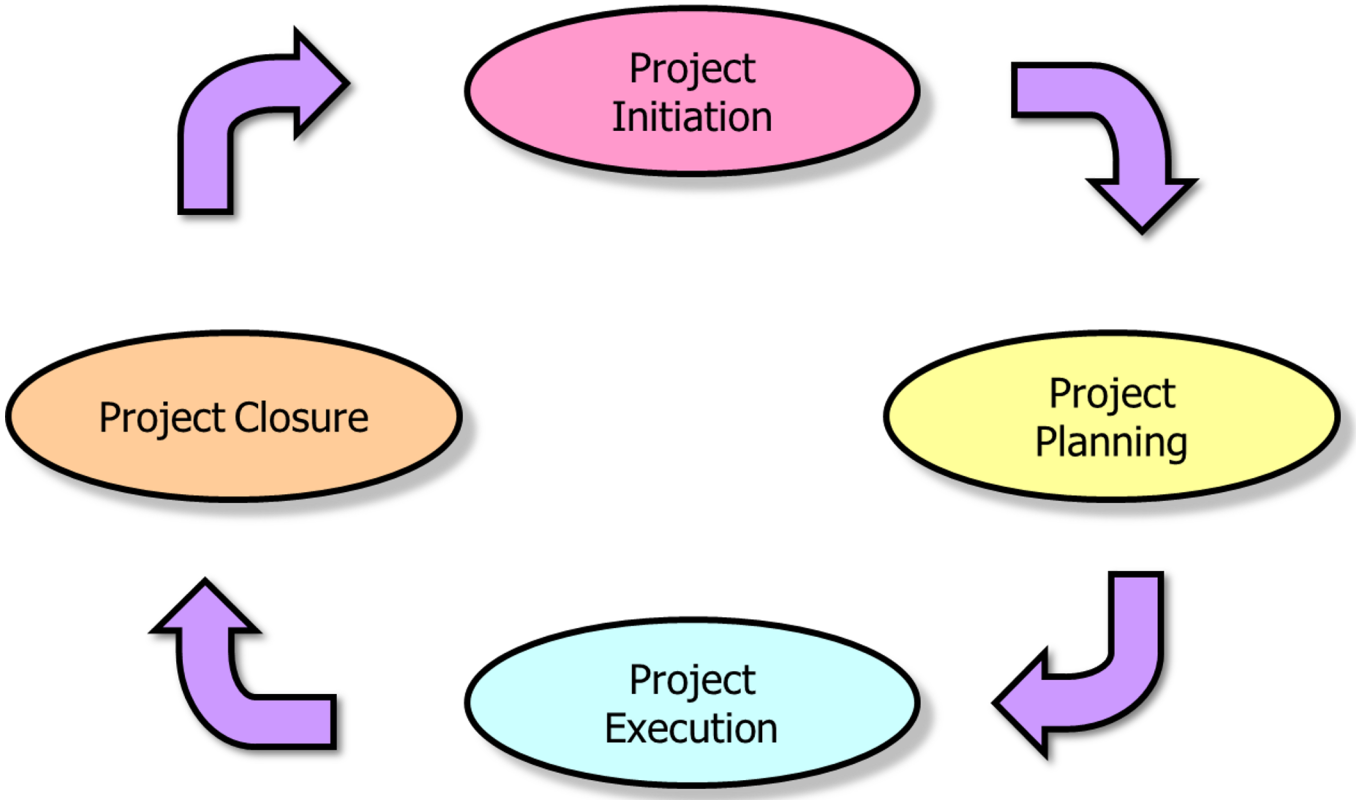
- Life cycle of a software project
- Role of project manager (PM)
- How to plan & schedule a project
- How to handle problems
- Quality management

Need Project Management?

- Standish Group's CHAOS research projects on 'IT project success rates and management best practices' say:
  - 31.1% of software projects will be canceled before they are completed
  - 52.7% of software projects will cost 189% of their original estimates
  - 57% of the projects that fail are due to communication breakdown
  - 32% of IT projects face challenges due to unclear objective and outcomes

Source: https://www.csus.edu/indiv/v/velianitis/161/chaosreport.pdf

Project Life Cycle



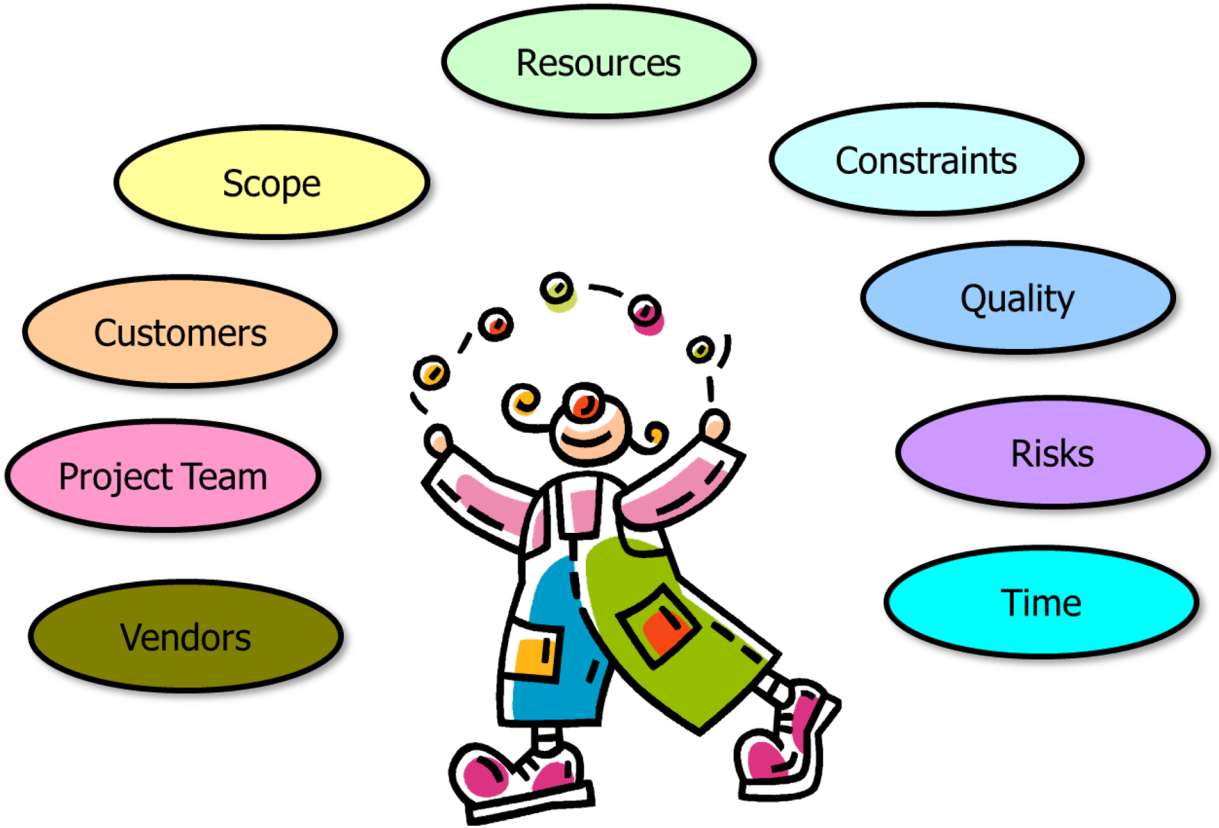
Project Life Cycle

- **Initiation**
  - Identify a business problem
  - Conduct feasibility study
  - Create Term of Reference (ToR)
- **Planning**
  - Create a plan to meet goals
    - “Right size” the life cycle
    - Estimate scope, effort and schedule
    - Choose metrics
    - Manage risks

Project Life Cycle

- **Execution**
  - Monitor status, progress and quality (through project metrics)
  - Update project plan
  - Keep all skate-holders up-to-date
- **Closure**
  - Close all problems and issues
  - Post-analysis of metrics, process, teamwork, communication
  - Document the lessons learned

Project Manager (PM)



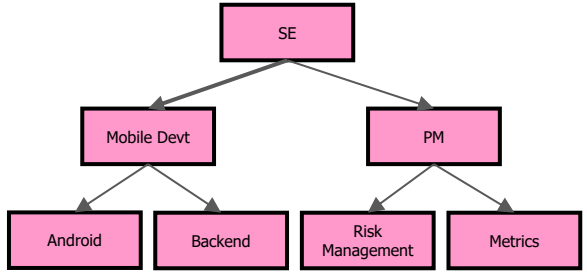
What to Manage

- Four essentials
  - Cost
  - Time
  - Scope
  - Quality

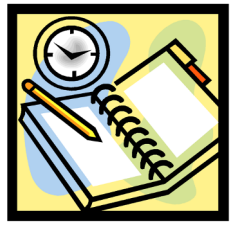
How to Plan

Step 1: Define Scope

Step 2: Decide Process



Step 3: Work Breakdown



Step 4: Create Schedule

Step 5: Estimate cost

Step 6: Analyze Risk

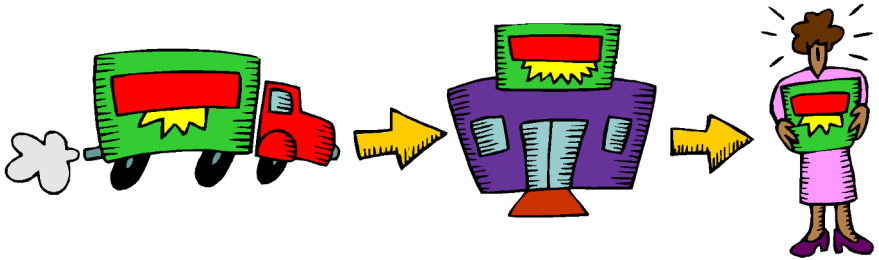
Step 1: Define Scope

- Project scope identifies and defines the tasks to be completed in order to meet the project objectives
- The list of project tasks is identified from the project requirements supplied by the key users

  - ❑ Functionality
  - ❑ Usability
  - ❑ Reliability
  - ❑ Performance
  - ❑ Supportability

Step 2: Decide Process

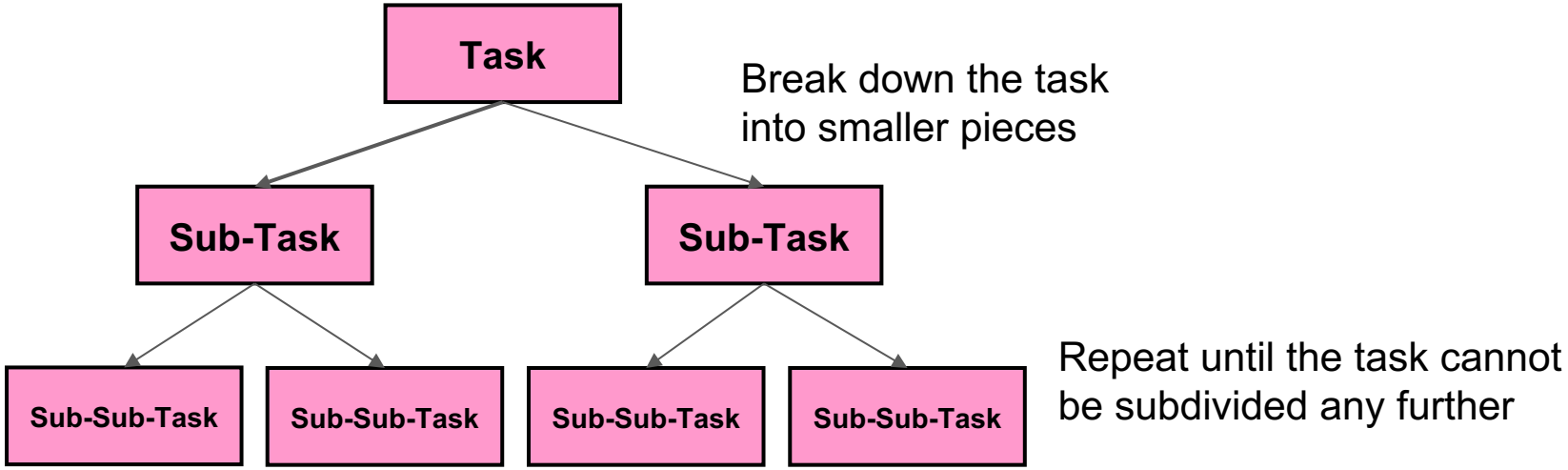
- Strength of process includes
  - Project size
  - Team distributions
  - Complexity of technology
  - Number of stakeholders
  - Compliance requirements
  - Where in the Project Lifecycle



- Project Process
  - Iterations for continuous progress
  - Tracking the schedule and metrics
  - Collective Code Ownership
  - Use Case-driven development
  - Coding Standards
  - Use of Git
  - Regular rotation of roles
  - Working demo per iteration
  - Unit and integration testing
  - etc.

Step 3: Work Breakdown

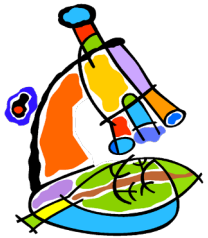
- Defining and organizing the total scope of the project
- Arranged in a hierarchical structure



Step 4: Create Schedule

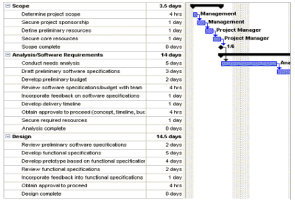
1. Define Activities

2. Sequence activities



3. Allocate Resources

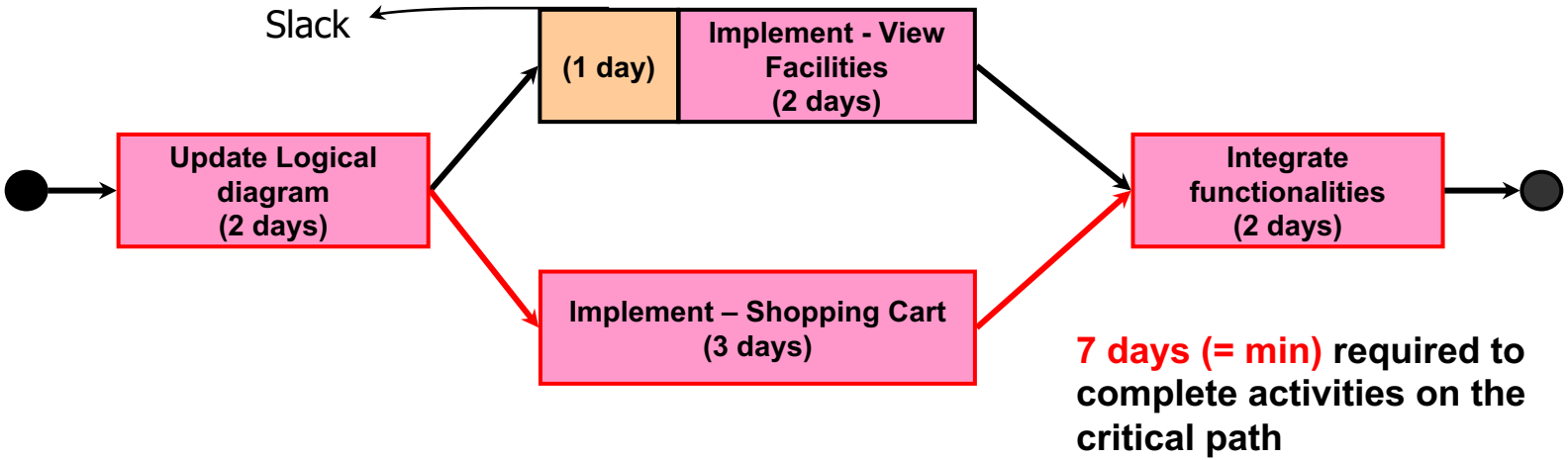
4. Estimate Effort



5. Develop Schedule

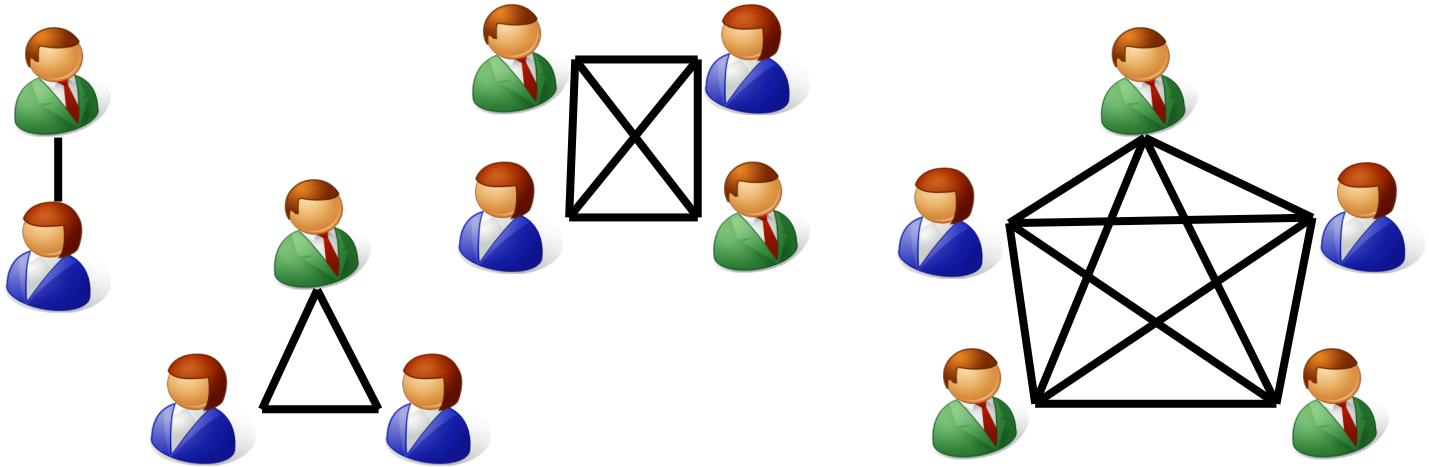
Sequence Activities: Critical Path

- A set of activities critical to the completion of project
  - Delay on an activity on the critical path impacts the completion



Allocate Resources: Staff

- Number of people required
- Match people's skills to the task
- Minimize conflicts
- Have a staff allocation chart

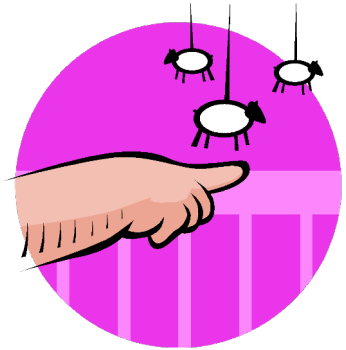


Allocate Resources: Equipments

- Try to allocate sensibly
  - Bar chart with the people named
  - Looking for gross allocation clashes
    - Time & Location should make sense
- Two types of dependencies
  - Logical / task dependencies
  - Staff dependencies
- Update resource allocations in light of experience
  - Changing schedule can overcome problems

Effort Estimation

- Analogous Estimating
  - Use actual duration of a previous, similar activity as the basis
- Expert Judgement
  - Wideband Delphi
    - Consensus-based estimation technique for estimating effort
- Use Case points



Effort into a Schedule

- Can either be
  - Optimistic (usually unrealistic)
  - Nominal (sometimes realistic)
  - Pessimistic (often realistic)
- Realistic Schedules allow for
  - Overhead, meetings, client issues
  - Learning, training, “down time”, vacations
  - Test planning, testing, documentation
  - People, process, technology problems, things going wrong

Developing Schedule

- Milestones
  - Endpoints of some activities
    - At some logical point. Not halfway through coding!
  - Needs to be checked
  - Should have start & end ‘ceremony’
    - Demo of build
    - Report to management
  - Avoid milestones being too frequent or infrequent
- Buffer
  - Individual task? Or to the whole project?

Accuracy vs. Cost

- Tausworthe's Principle
  - As the number of milestones increases, so does the accuracy of the estimate
- Overhead
  - "Checking" process
  - Each milestone should have start & end "ceremony"
    - a demo, report to a boss, etc.

Handling Schedule Problems

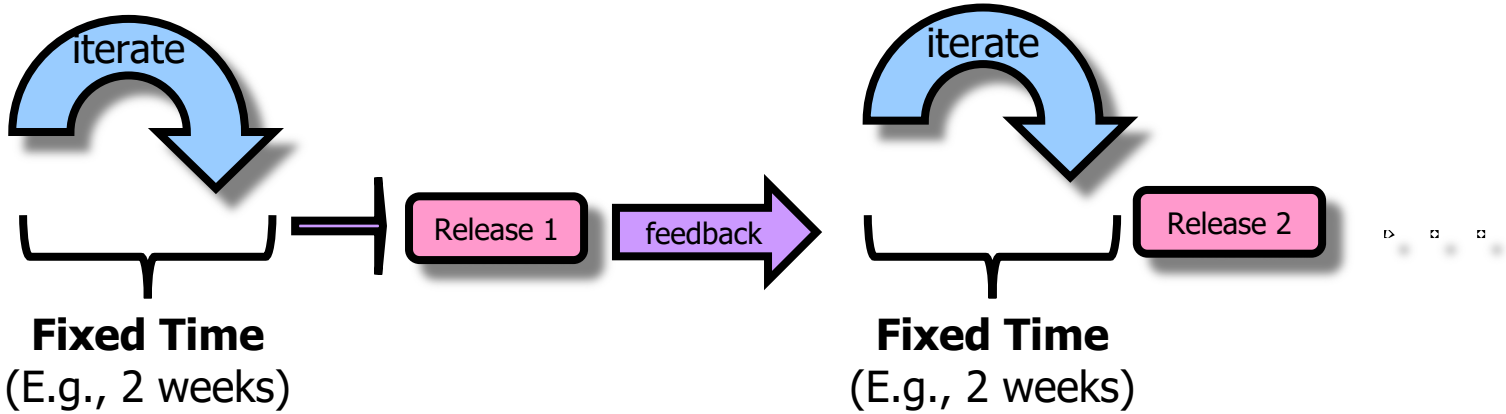
- Work Harder?
  - Breaks down quickly
- Hire more people?
  - Brooke's Law
    - Adding manpower to a late software project makes it later
    - Catch-up time, plus more overhead often cancel the effect of extra resources
- Re-Scope
  - Admit "defeat". Be more sensible
  - Better to have something than nothing

Bad Scheduling/Estimation

- Estimates are never updated and improved from the previous iteration
- Tasks that are too long a duration (15 days)
- Tasks that are too general (Task Coding)
- Tasks that has nobody assigned to it
  - Or the “team” is assigned to it
  - Variant: small set of people assigned to most of the tasks
- Same (generic) tasks for all iterations
  - Nobody knows what is supposed to be done
  - “Code”, “update use case”, etc.
- Schedule done on a week basis / not a task basis
  - Week 1: stuff, Week 2: same stuff
  - You are not hourly workers!

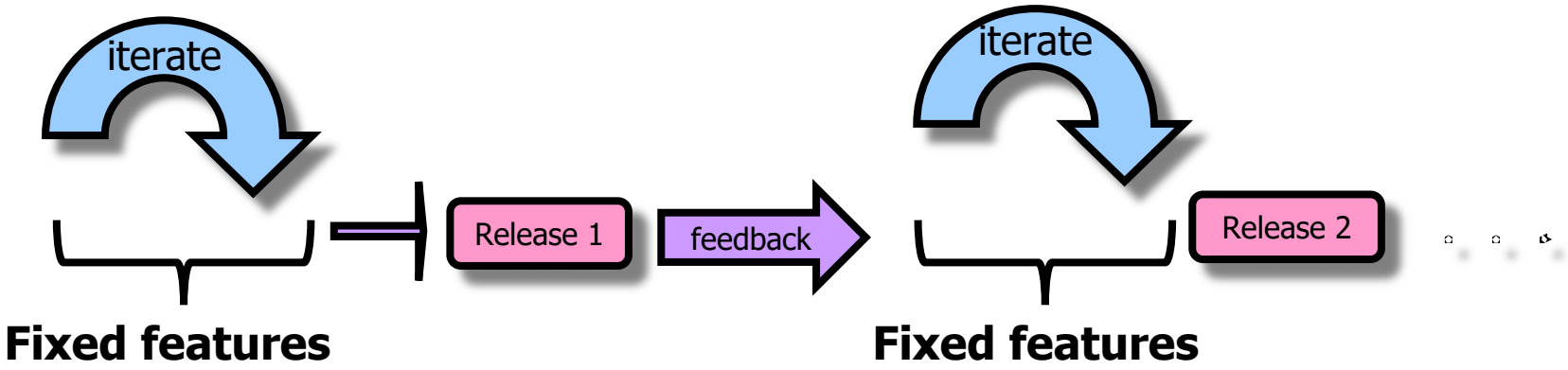
Iterative Time Boxing

- Iterate in **fixed** length of **time**
  - OK to slip some functionality
  - Not OK to slip the end of date of an iteration
  - Figure out what the real requirements are
  - Build-Feedback-Adapt cycles
    - Each cycle increase scope/functionality

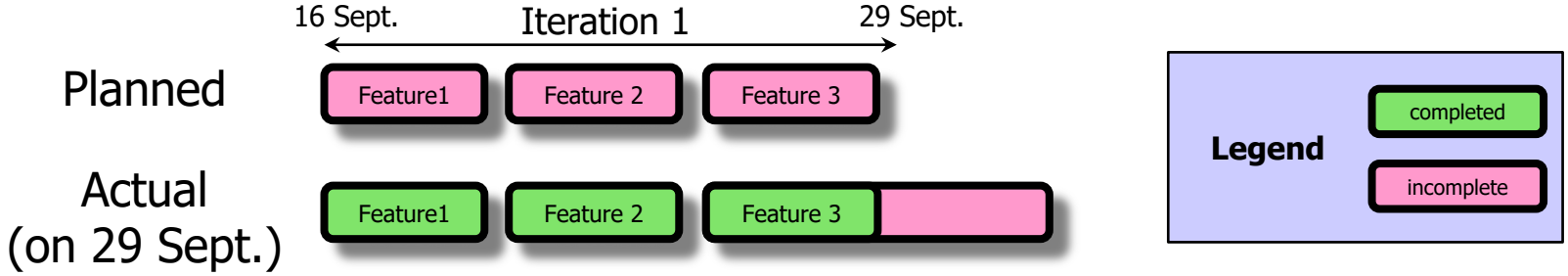


Iterative Feature Boxing

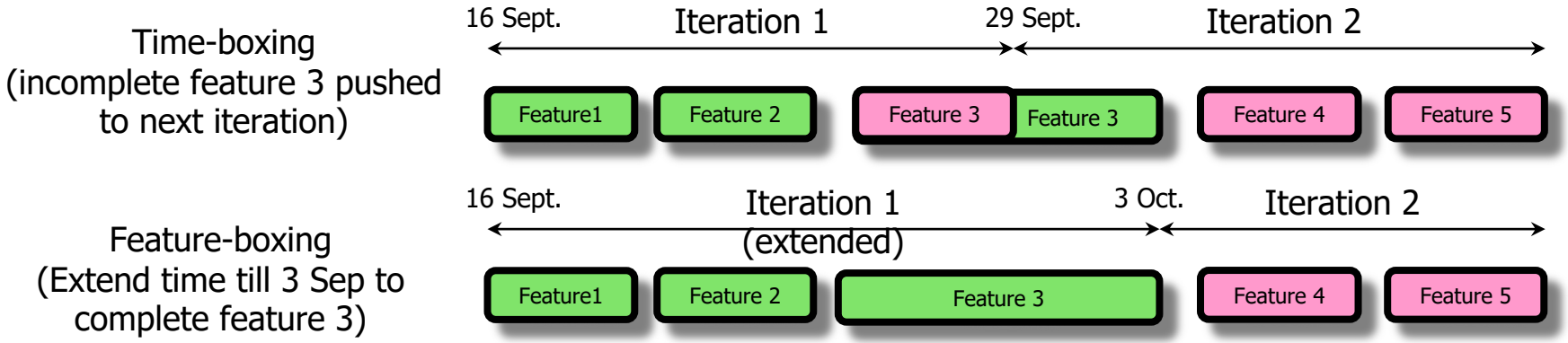
- Iterated with **fixed** number of **features**
  - Slip the end date of an iteration to complete all features



Time Boxing vs. Feature Boxing



Managing the Overrun using Time-boxing and Feature Boxing



What should you use for the Project?

- You can only use **time boxing**
  - Time boxing: Unfinished functionality is moved to a future iteration. Next iteration starts on time
  - Feature boxing: Next iteration starts late

- Use a task metric to track the percentage of tasks completed vs. planned.

Scheduling Guides for Project (1/3)

- Your schedule must be created until the end of the project
  - Think through as much as possible though it can flexibly change
- Anything needed for your project should be scheduled
  - E.g. internal meetings, testing, debugging, coaching
  - Omit team bonding, lunch, dinner, etc.
- Every iteration should include iteration kick-off meeting
  - PM should update the schedule for the corresponding iteration after the kick-off meeting

Scheduling Guides for Project (2/3)

- An implementation task should be followed by a git push to a shared github repository
- PM should update the documents
  - Initial creation in iteration 1 should be done together

Scheduling Guides for Project (3/3)

- (From iteration 1) Feature implementation should be done in a separate branch and must push the code. Leave pull-request (PR) if necessary
  - PM should merge PRs (details discussed in week 4)
- (From iteration 3) programmers will perform unit test before requesting merge
- (From iteration 3) PM should perform integration tests with the **features from the current iteration**

Scheduling Tips for Project (1/2)

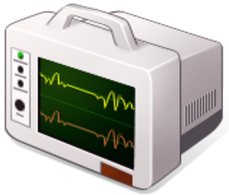
- Don’t forget to update the schedule and document wiki
- We strongly recommend iteration review meetings
  - Was the scheduling okay, too much, too little?
  - Code sharing could be driven by the strongest technical lead
- Do not neglect kickoff and review meetings
  - Very good time to sync as a whole team
  - Lessons learned from each iteration improve the future iterations

Scheduling Tips for Project (2/2)

- Highly recommend PMs to organize feedbacks and issues for good management
  - Also helps handing over to next PM
- Do not hesitate to ask questions to the team (and TAs!)
  - Be open to spend time for communication
  - Helping others will eventually help yourself

Step 5: Estimate Cost

- Determine Resource Cost Rates
- Vendor Bid Analysis
- Cost of Quality



Step 6: Identify Risk

- A risk is an event or condition that may occur, and whose occurrence, if it does not take place, has a harmful or negative effect on a project
- Risk exposure
  - **Probability** of occurrence X **Impact** on the project

Risk Management Strategies

Strategies	Description
Risk avoidance	Reorganize the project plan to eliminate the risk.
Risk mitigation	Reduce the probability or impact of the risk to an acceptable level. Examples of mitigation actions include scope reduction, additional tests, incorporating redundancy into the system.
Risk transfer	Move the responsibility of the potential risk to a third-party. This may be in the form of subcontracting a piece of task that the team has insufficient skill-set to handle to the third-party for a premium on the risk.
Risk acceptance	Live with it.
Contingency Plan	Plan “Plan B”. “Plan B” normally kicks in when the risk is about to materialize or has occurred.

Risk Assessment

- ‘A’ risks need the most attention and most well developed mitigation or recovery strategies
- ‘C’ risks can occur but deserve the least amount of planning

		Likelihood		
		Low	Medium	High
Impact	High	B	A	A
	Medium	C	B	A
	Low	C	C	B

Risk Table - Good Example

S/N	Risk Statement	Consequence	Likelihood (H/M/L)	Impact (H/M/L)	Level (Derived)	Mitigation Strategy &/or Contingency Plan
1	Team is unfamiliar with Android platform	(a) Project delays due to incorrect estimates (b) Lower quality as there will be more bugs in the system	High	High	A	1. Development team to be trained in new platform by Sep 2025. 2. Procure additional support of tool for the first month of development phase.

Risk Table - Bad Example

S/N	Risk Statement	Consequence	Likelihood (H/M/L)	Impact (H/M/L)	Level (Derived)	Mitigation Strategy &/or Contingency Plan
1	Changes in work schedule due to unexpected events; delay of first release and other milestones	Project will be delayed	High	High	A	Raise awareness about the change and re-look the work breakdown structure and task allocations.

Worse Risks

- Overlapping of work done by team members
- Different coding style by each team member
- Nobody knows who has the latest copy of the code
- Members not delivering tasks assigned on time, delaying the schedule
- Don't meet requirements due to lack of planning
- Cannot complete on time due to too big a scope
- System crash due to poor maintenance
- Out of scope/digression because team never follows plan
- Irrelevance caused by digression
- Fails to manage risks after submitting project plan
- No updating of the list

Summary

- A project cycle includes initiation, planning, execution and closure
- Planning should provide a project schedule, cost estimation and risk analysis for the execution stage
- Good schedules can be executed with minimal corrections

Team Exercises

10 mins

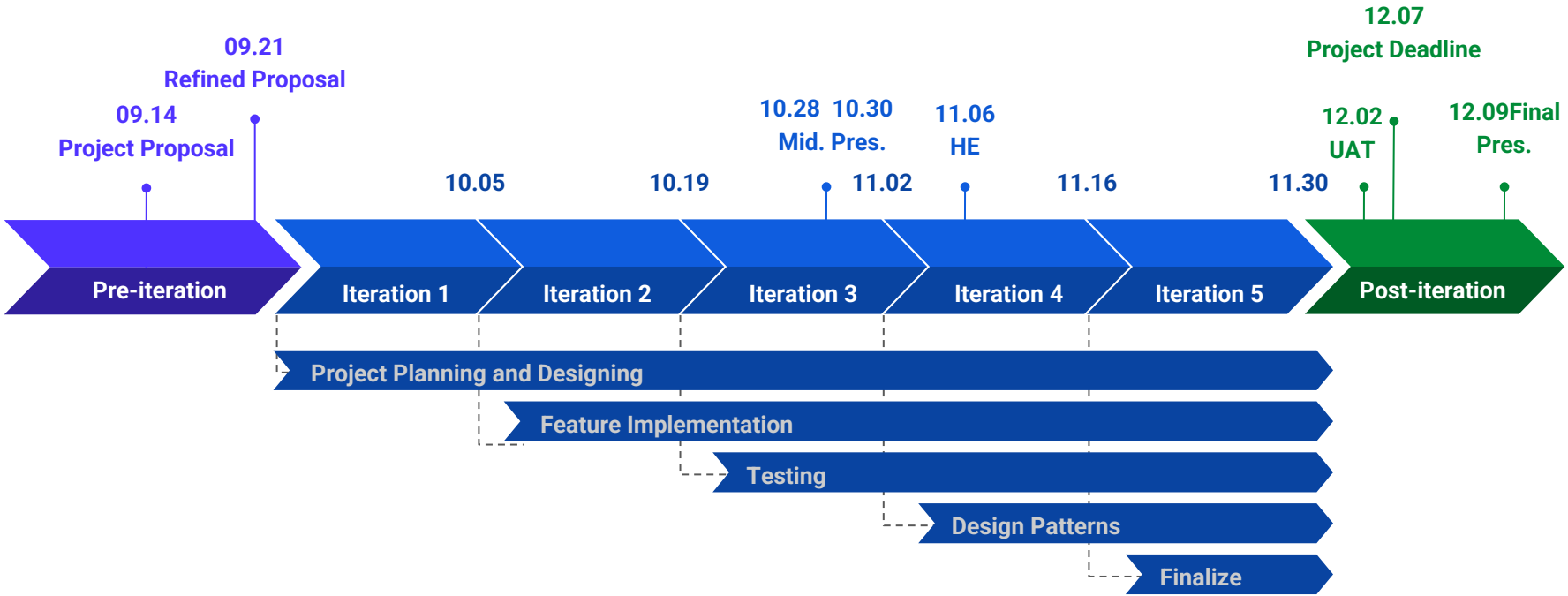
Team Exercise 1: Role Rotation (1/3)

- Decide the roles for all 5 iterations
- PM is going to be busy
- Rule of thumb
  - The role rotates every iteration and stays the same during the iteration
  - Everyone needs to participate as a PM for an iteration
  - Other 4 participants will contribute to develop

10 mins

Team Exercise 1: Role Rotation (2/3)

- Understand what to be done for each iteration



10 mins

Team Exercise 1: Role Rotation (3/3)

- Copy the Project Schedule Template, and fill in the roles in the ‘Overview’ tab.
  - You will use the spreadsheet over the project, so be sure to have it shared with the team
- Submit your spreadsheet link via ETL

25 mins

Team Exercise 2: Refine Ideas

- You will refine the ideas for the project proposed
- Here are some guidelines
  - Accept all the wild ideas (creativity over feasibility)
  - Build on others' ideas (don't criticize)
  - Aim quantity over quality
  - Make the session visual
    - You will use the template we provide for today
  - Try to make the session collaborative and fun