Project Management (Part 2)

Week 3

Do what you love, and do it well – that's much more meaningful than any metric
— Kevin Systrom

Human-Centered Computer Systems Lab

SEOUL NATIONAL UNIVERSITY

RECAP

Project Life Cycle

1. Initiation
2. Planning
3. Execution
4. Closure

How to Plan

1. Define scope
2. Decide process
3. Define work breakdown structure
4. Create schedule
5. Estimate cost
6. Analyze risk

How to Create Schedule

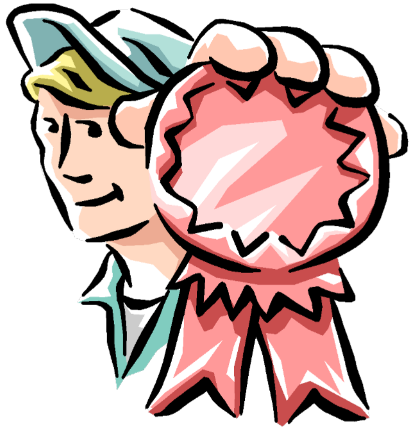
1. Define activities
2. Sequence activities
3. Allocate resources
4. Estimate duration
5. Develop schedule

Quality Management

- What is quality?
  - Achieving agreed requirements
  - Getting it right the first time
- Quality Assurance
  - Establish project standards and procedures
- Quality Control
  - Inspection of deliverables for defects

Good Process

- We want the process to
  - Become repeatable
  - Be well defined
  - Get more reliable
  - Get more efficient

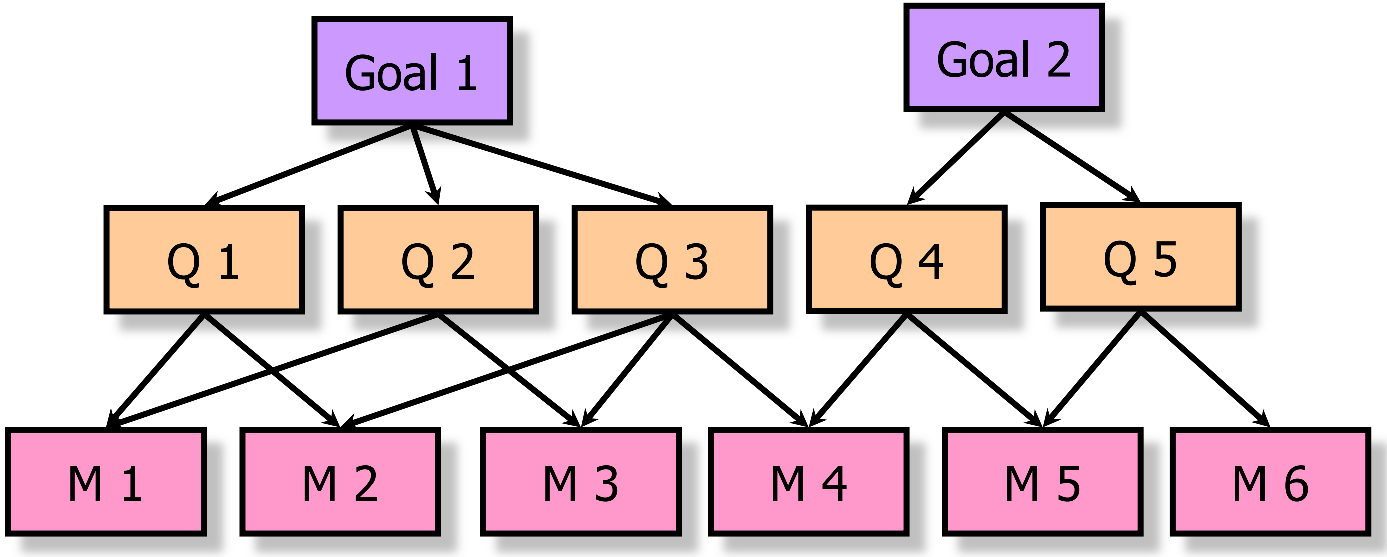


Metrics: Introduction

- Metrics are simply numbers to help you evaluate your process
  - They should help you. Not delay you
- GQM is a goal-oriented approach to design software metrics
  - Goal: what should be improved
  - Question: what details can we ask about the process
  - Metric: what numbers can we put on the question

Goal-Question-Metric (GQM)

- A top-down method for creating a metrics program. Good for aligning measurements with business goals



GQM Step 1: Select a Project Goal

- State the goal as quantitative and measurable as possible
- Examples of goals
    - Improve software quality
    - Increase percentage of on schedule projects to 80%
    - Reduce maintenance costs by 25%
    - Improve schedule estimation accuracy to within 10% actual
    - Reduce system testing time by three weeks on the next project

GQM Step 2: For Each Goal, Ask Questions

- Ask questions that tell more about the progress towards the goal
- Examples
  - Goal
    - Improve software quality
  - Questions
    - How many bugs do we have?
    - How well understood is the documentation?
    - Are many changes required?
    - Do we reuse reliable code?

GQM Step 3: Identify Measures / Metrics

- Identify measures or metrics to address each question
  - Measures: can be directly measures/recorded
  - Metrics: values computed from two or more measures
- Examples
  - Total bugs found before release / product size
  - Number of bugs reported post delivery
  - Number of clarification calls received by help desk
  - Open bug at release / product size
  - Average time from bug report to closure
  - What percentage of the new product’s code is from reused modules?

GQM Step 4: Improve the Process

- What part of the process will be changed
- Examples
  - Increase number of test cases if there are too many defects
  - Allocate a “bug squashing day” each week / iteration / month depending on bug rate
  - Institute maximum bug age policy:
    - require owner to fix all defects > x days old
  - Require code review on a module over normal defect rate

GQM Example 1

- Goal:
    - Deliver when we say we will
- Question:
    - How accurate are our estimates?
- Metric:
    - Estimated time / actual time
- Use:
    - Adjust future schedule using metric ratio
    - Use metric to highlight estimation errors

GQM Example 2

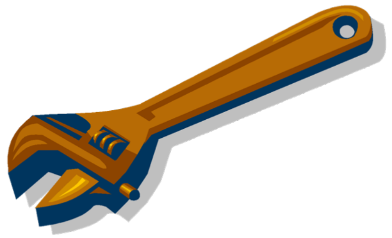
- Goal:
  - Improve code quality
- Question:
  - How buggy is our code?
- Metric:
  - Number of bugs found throughout the project
  - Average number of bugs in each web page or class
- Use:
  - 1: Raise awareness of current quality
  - 2: Identify problem areas (>average bugs)

Measurements

- Resistance to measurement programs
  - Time consuming to collect data
  - Irrelevant to building the right product
  - Might be used against them
- Path to success
  - Start small
  - Visible use of data
  - Don’t use measurements to evaluate individuals

Measurements: How

- Use tools
  - SLOC tracking
  - Bug tracking
- Simple tracking
  - Low overhead, easy to compile
- Use charts
  - Make it visible
- Make it a habit
  - Consistent and easy



Metrics to Consider

- Estimated and actual task duration
- Work effort duration
- Defects
- Product size
- Product quality
- Process quality
- Project status
- Consumer satisfaction

Metrics for Project

- Your team is supposed to track the following metrics:
  - Task Metric (TM)
  - Programming Hours Metrics (PHM)
  - Load Factor
  - Hours per Member
- These metrics are tracked with the template provided automatically

Task Metric (TM)

- Formula:
  - Actual tasks completed / Estimated tasks
- Tracked on per iteration basis
- The estimated and actual number of tasks includes every task (not just programming tasks) in the schedule for that iteration

Task Metric (TM)

Score(%)	Action
TM < 50	1. Inform your TA about the slip. 2. Re-estimate the tasks for the future iterations. 3. Seriously consider dropping tasks.
50 < TM <= 90	1. Re-estimate the tasks for the future iterations. 2. If there is no time to recover, decide the functionalities to drop.
90 < TM <= 110	1. Our estimates are fairly accurate, and we are roughly on track.
110 < TM <= 150	1. Gross over-estimated the effort required. 2. Re-estimate the tasks for the future iterations.
150 < TM	1. Inform your supervisor about the slip. 2. Re-estimate the tasks for the future iterations. 3. Seriously consider adding tasks.

Schedule Metric (SM) for Feature Boxing

- Formula:
  - Estimated time / actual time
- Tracked on per Iteration basis
- The estimated and actual time taken must apply to the critical path
  - Not merely just the sum of all the tasks' time
- The critical path must include all activities in the iteration (integration, testing, debugging)
  - Buffer time is not included!

Schedule Metric (SM) for Feature Boxing

Score(%)	Action
SM < 50	1. Inform your supervisor about the slip. 2. Then use the same mitigation as the category 50 < TM < 90.
50 < SM <= 90	1. Re-estimate the tasks for the future iterations. 2. Deduct the number of days behind schedule from buffer days. 3. If there is no more buffer day, decide the functionalities to drop.
90 < SM <= 110	Our estimates are fairly accurate, and we are roughly on track. 1. Add/Deduct the number of days behind schedule from buffer days. 2. If there is no more buffer day, decide the functionalities to drop
110 < SM <= 150	Gross over-estimated the effort required. 1. Re-estimate the tasks for the future iterations. 2. Add the number of days gained to buffer days.
150 < SM	1. Inform your supervisor about the slip. 2. Then use the same mitigation as the category 110 < TM < 150

Programming Hours Metrics (PHM)

- Formula:
  - Estimated programming hours / actual programming hours (per task)
- A task must be in the schedule. Each task can be
  - Implementation of a use case (UI to backend)
  - A set of data managers & entities
  - Bug fixing, etc.

Score(%)	Action
PHM < 50	1. Re-estimate the tasks in the coming iteration(s) and update the schedule document. 2. Review possible estimation issues.
50 < PHM <= 150	No action required.
150 < PHM	1. Re-estimate the tasks in the coming iteration(s)) and update the schedule document. 2. Review possible estimation issues.

Load Factor

- Formula:
    - Estimated Hours / Actual Hours
- Tracked on per iteration basis
- vs. PHM
    - Every task (not just programming tasks) in the schedule affects load factor
    - PHM is tracked per task, while load factor is tracked per iteration to check the quality of schedule of the iteration

Hours per Member

- Formula:
    - Programming (and Non-programming) hours per member
- Enables easy comparison between the hours spent by the teammates
- Please try to make it roughly even for the project

Example

- TA Project Schedule
    - Check the ‘Metrics’ tab

Summary

- Good processes are well defined, reliable, efficient and can be repeated
- Using metrics is a good way to manage quality
    - GQM helps deciding on the metric
- We will track 5 metrics for the project

Useful References for PM

- A Software Metrics Primer
- Implementing a Successful Measurement Program
- CMU Software Metrics
- Experiences in Implementing Measurement Programs
- Planning Poker
- Use Case 1, Use Case 2
- Evidence Based Scheduling
- Building a Fort
- IBM On Climbing Big Mountains
- Know Your Enemy: Introduction to Risk Management

Team Exercises

40 mins

Team Exercise 1: Listing Tasks (1/4)

- We will breakdown the tasks before scheduling
  - List of description of the task + estimated hours
- The list should include all the tasks necessary
  - Following the schedule generated from the list should complete the project

40 mins

Team Exercise 1: Listing Tasks (2/4)

- PM tasks
  - Updates (what should PM update?)
  - Reviews (what should PM review?)
- Team tasks
  - Periodic meetings
    - You must include daily meetups, iteration kickoff/review meetings
  - Aperiodic sessions (e.g., library learning, tool learning, etc.)
- Design tasks
  - The contents of the design document
  - Large portion should be done in iteration 1

40 mins

Team Exercise 1: Listing Tasks (3/4)

- Implementation & testing tasks
  - Major part of task breakdown
  - Start from listing the primary features
    - Breakdown into unit tasks & tests
  - Plan for the integration tests
- Milestones (and its preparations)
  - Midterm presentation in iteration 3
  - Heuristic evaluation in iteration 4

40 mins

Team Exercise 1: Listing Tasks (4/4)

- Others
  - Anything else your team wants that do not fit in the previous categories
  - Only project-related tasks should be scheduled
- Use the following template
- Submit the pdf file via ETL (9/18 23:59)
  - File name: team{XX}_tasks.pdf
  - Make sure anyone with the link can view the document

20 mins

Team Exercise 2: Scheduling Iter 1 (1/6)

- We will schedule the first iteration with the tasks from exercise 1
- Please use the following schedule template
- You can refer to the example from TAs

20 mins

Team Exercise 2: Scheduling Iter 1 (2/6)

- ‘Overview’ tab
  - Includes ‘Team Information’, ‘Project Overview’ and ‘House Rules’
  - You may fill in this tab after the proposal is finalized
  - Please have it finalized at iteration 1 kick-off meeting

20 mins

Team Exercise 2: Scheduling Iter 1 (3/6)

- ‘Schedule’ tab
    - ‘Iteration No.’: iteration number
    - ‘Task ID’: {P or A}{X}
        - ‘P’ if the schedule planned in the beginning of the iteration
        - ‘A’ if the schedule is added during the iteration
        - X denotes X-th (planned or added) tasks
    - ‘Description’: description of the task
        - Every team member should understand what the task is about just by reading the description

20 mins

Team Exercise 2: Scheduling Iter 1 (4/6)

- ‘Schedule’ tab
  - ‘Type’: type of the tasks
    - Ask TA if the task does not seem fit in the provided categories
  - ‘Deadline’, ‘Planned Hours’, ‘Actual End Date’, ‘Actual Hours Spent’: as it says
  - ‘Status’: shows the state of the task
    - ‘Started’ as you start (or re-start) the task
    - ‘Completed’ if finished
    - When you are not on the task currently,
      - ‘Incompleted’ if you are planning to finish the task
      - ‘Hold’ if it is unclear whether the task will be re-started

20 mins

Team Exercise 2: Scheduling Iter 1 (5/6)

- ‘Schedule’ tab
  - ‘Plan Unexpected’: mark yes if it is
    - Used to measure TM
  - ‘Final Commit ID’: git commit ID
    - Mandatory if the task requires any commit

20 mins

Team Exercise 2: Scheduling Iter 1 (6/6)

- Schedule iteration 1
  - Should include any tasks necessary for writing design documents, requirements and specifications, demo preparation, and initial scheduling of the whole project
  - Example from TAs are only a simple version - do not follow task-wise
- Review if the tasks listed in Exercise 1 is valid
- Submit the spreadsheet link via ETL

Note: Device Rental

- We will rent two smartphones to each team
  - State clearly in the final proposal if your team needs smartwatches or other devices
  - Do not lose or break the devices
- Procedure?
  - Find your TA after the next Tuesday class
  - Sign up for the rental sheet
  - Get the devices
  - Use them for development and testing
  - Return by the end of the semester. More details to come