Prifysgol Abertawe
Swansea University

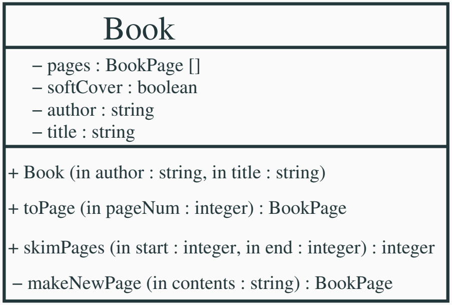
CS-230 Software Engineering

L04: Team Roles, Project Management & Minutes of Meeting

Dr. Liam O'Reilly

Semester 1 – 2020

Previously in CS 230...



```
Book
– pages : BookPage []
– softCover : boolean
– author : string
– title : string
+ Book (in author : string, in title : string)
+ toPage (in pageNum : integer) : BookPage
+ skimPages (in start : integer, in end : integer) : integer
– makeNewPage (in contents : string) : BookPage
```

Design... No Lumping Together!!!

Facebook Illusion

- Who thinks this:

- ... was programmed by a single person?
- At one point, it may have been but now it isn't.
- This is exactly why a group work based software engineering module is needed!

Properties of a Good Team (In My Opinion...)

- Everyone contributes actively to the work (for CS-230).
- Clear division of labour to avoid work duplication.
  - You know where you fit in to the larger project.
  - You understand your responsibilities.
- Good and clear communication.
  - Leader(s) help foster a clear direction.
- Members complement each other's weaknesses.
- Spend more time listening and less time talking.
  - Always be aware of people that aren't speaking.
  - Offer opportunities for them to contribute.
- Everyone feels at any time that they can contribute.

Roles and Responsibilities in a Team

- Each member of each group has responsibilities.
- In software engineering, the following roles are somewhat typical:
  - Customer Interface Manager
  - Design Manager
  - Implementation Manager
  - Test Manager
  - Planning and Quality Manager

Customer Interface Manager

- Customer interface manger is responsible for:
  1. Team's relationships with its customer(s).
  2. Clearing up ambiguities in requirements specification.

- Some questions that the customer interface manager addresses are:
  - Are we being responsive to customer requests?
  - Is quality of requirements sufficient to guide development?
  - Do all team members understand requirements?

Design Manager

- Responsible for coordinating the team's design documentation.
- Tracks that design documentation standards are being met and recorded.
- Some questions that design manager addresses are:
    - Do all team members understand how to read/write UML?
    - Does everyone understand the design methodology?
    - Is the team's design work of high quality?
    - Does software design consider future product evolution?
    - Is design properly documented?
- In A1 and A2, everyone contributes to the design and code.
- However, someone should help coordinate it.

Implementation Manager

- Responsible for quality of team's implementation.
- Some questions implementation manager addresses are:
    - How strong are the programmers in this group?
        - If some members are weak, encourage them to catch up.
        - Everyone should have a programming reference book.
    - Can the design be clearer.
    - Does the code follow the coding standards.
    - Does do the bits of code integrate well into the entire system?.
- In A1 and A2, everyone contributes to the implementation
- However, someone should help coordinate it

Test Manager

- Responsible the quality of team's testing and test-related work
- Some questions the test manager addresses are:
  - Are test plans produced when the process needs them?
  - Do all classes have unit tests defined?
  - Are all methods tested by unit tests?
  - Are test plans complete and thorough?
  - Does each team member understand how to produce tests?
  - What is team's integration test strategy?
- For practical reasons, A1 and A2 do not contain too much testing.
  - Very very important though and more of it in CS-235.
- We will look at testing later in this course, but you have already done a lot last year in in CS-135.

Planning and Quality Manager

- Responsible for team's plans, reporting, plan status.
- Some questions support manager addresses are:
    - Is team meeting often enough?
    - Is team following minutes protocol?
    - Does team want to use version control?
    - Are code inspections being carried out?
    - Is implementation properly documented using Javadoc/?Doxygen?
    - Do quality of all classes and code meet team's quality standards?
        - If not, what is recommended to fix this.
    - Overall coordination of the development procedure.

Teamwork and Management for CS-230

- In general, management roles should be covered by team members.
- Each team member contributes to all phases of software life-cycle.
    - E.g., Implementation manger ensures that each team member contributes to implementation.

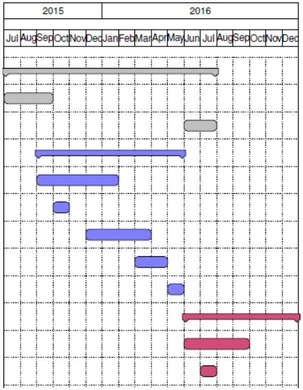
Advantages of Small Teams

- Planning, communication, and coordination is simplified. The larger the team, the more complex.
- Meetings are easier to plan.
- Responsibilities are easier to track, less diffusion of responsibility.
- Everyone learns more.
- However, we need to ensure that everyone contributes...

Project Planning

- You have probably chosen a software engineering model
    - Waterfall
    - Scrum agile
    - etc.
- Given that model, break the work down and manage the schedule.
    - Scrum? Where are your sprints? Where are your meetings?
    - Extreme Programming? Where are your code reviews?

Gantt Chart



- A Gantt chart can help you organise this information.
- Most projects require this road-map.

Creating Gantt Charts

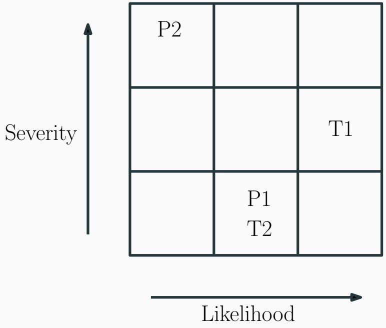
1. Break your project up into smaller tasks.
2. Find obvious milestones (points that you reach).
3. Workout the dependencies between tasks, which need to be done before others can start.
4. Estimate duration of each tasks. Try to be realistic.
5. You can now put it all together.

Project Risks

- With every project there are risks.
- Personal risks - risk involving yourself or your actions:
    - Illness, unable to understand some material.
    - Hard drive crash and no backups.
- Technical risks - intrinsically hard because of the problem:
    - A particular algorithm is very difficult to implement.
    - Interfaces between many different software/hardware components tricky.
- Each risk should have a description and a mitigation strategy:
    - describe what the risk is.
    - describe what you will do about it if it happens.

Severity and Likelihood

- Each risk also has a severity and a likelihood:
  - Severity - If it happens, what is the impact on the project?
  - Likelihood - How likely is it to happen?
- Both can be represented compactly as a table.



You should be most concerned with Top-right area, less concerned with bottom-left.

Advice for CS-230 Group Work

Practicalities of Group Work in the Course

- Your groups are carefully created by me.
    - Partially random, with large influence of previous grades.
    - Each group has the same "average" mark for the first year (to within a few percent).
- You might not be with your friends.
- There may be issues with other members.
- Alterations for groups will not be considered.

You Non-Contributors!

- *Team member X does nothing! X gets free marks off of our work!*
  - You could be the problem (let's hope so. You can fix it!)
  - Create an atmosphere where everyone generally feels that they can contribute positively.
  - Encourage them to ask questions. Ask them what they think.
  - Spend more time listening to them.
  - Remind them that they are to be involved in both design and implementation.
- If they are weak programmers, value non-technical contributions.
  - They are responsible for learning how the technical stuff works.
  - Spend a bit of time helping them learn.
    - A bit of time now will save you heaps of time later!

In the Real World...

- *In the Real World, X would be fired! This is totally unrealistic!*
  - It usually takes a bit (a lot) of time before that can happen.
  - Not enough time in two terms.
  - Process is not so fast.
- You might even have to deal with this in the Real World...

Dealing with Non-Contributors

- We've had many complaints about non-Contributors in the past.
- Yet, there are no "lone wolves" in software engineering.
  - Software in the Real World is developed in teams!
- To eliminate, even decrease importance of, group work in software engineering would do you a disservice to you!
- Still, the above complaints are the most frequent that the module has received over the years.

Dealing with Non-Contributors (2)

- To explicitly deal with non-Contributors, We've implemented the following:
  - Contribution Breakdowns that weight A1 and A2 marks.
- These contribution breakdowns (i.e., coins) are designed to encourage participation from everyone.
- There is also the exam which is individual.

Dealing with Non-Contributors (3)

- If a member still does not contribute, despite incentives:
    - as early as possible, notify your Academic Mentor.
    - Work with your Academic Mentor to encourage participation.
        - Be only positive – never negative or aggressive.
    - Keep the Academic Mentor posted on the situation.

Minutes of Meeting for CS-230

Thanks to Robert Laramee, much of this is based from Bob's Minutes of Meeting Protocol

Main Idea of Minutes of Meetings

When you hold a formal meeting (e.g, Contribution Breakdown Meetings), you should have a formal record of the meeting.

They provide the following benefits:

- Structure.
- Continuity and Memory.
- Aid in Progress.

Some one takes notes during the meeting and writes up the minutes afterwards. **They are not a transcript.** They capture only the main points.

Structure

They provide every meeting with structure and time itself is organised in a structured way.

Minutes are organised into three parts:

1. Topics discussed within the meeting and decisions that have been made.
2. Progress since the previous meeting
3. A Todo list of goals (actions) to achieve before the next meeting (assigned to individuals).

These above parts even help manage time during a meeting.

Continuity and Memory

Minutes provide continuity over time.

Each meeting starts out by reviewing the previous minutes of meeting.

This helps attendants remember precisely what topics where discussed at the last meeting.

People tend to forget what was discussed at the last meeting because they have so many other things going on

Aid in Progress

Since each minutes has a todo list of goals (assigned to individuals) to be achieved before the next meeting, the progress of every project can be monitored in structured fashion.

Also, larger tasks that may require months (or years) of work are broken down into smaller units of time which are much more manageable.

Attendants at meetings are more-or-less forced to decide what can be done between the current and next meetings.

Contents of Typical Minutes

Minutes should contain the following information:

1. The date of the meeting, it's starting time, and the ending time.
2. The name of each person present at the meeting.
3. A list of topics that were discussed at the meeting and any decisions that were made.
4. A list of progress, i.e., the things that were accomplished (and not accomplished) since the last meeting.
5. A todo list, i.e., a list of things, **assigned to individuals**, that are to be worked on prior to the next meeting.
6. Special Items CS-230 – See next slide.
7. The date and time of the next meeting.

Contents of Typical Minutes: CS-230 Extras

For CS-230 there are a few special things that need to be included too. See Assignment Overview for full details.

Main points: For unequal Contribution Breakdowns, the minutes must include:

- A clear description of why members distributed the way they did; and why members got the scores they did.
- What members with low payments can do next week to return to the expected equal division.

Length of Minutes

Minutes are not a transcript of the meeting.

For a 1 hour meeting the minutes should (probably) fit on a double side of A4.

Summary

We have looked at

- Software development teams.
- Typical roles in development teams.
- Planning - Gantt Charts.
- Risk Analysis.
- Advice for working in CS-230 groups.
- Minutes of Meetings