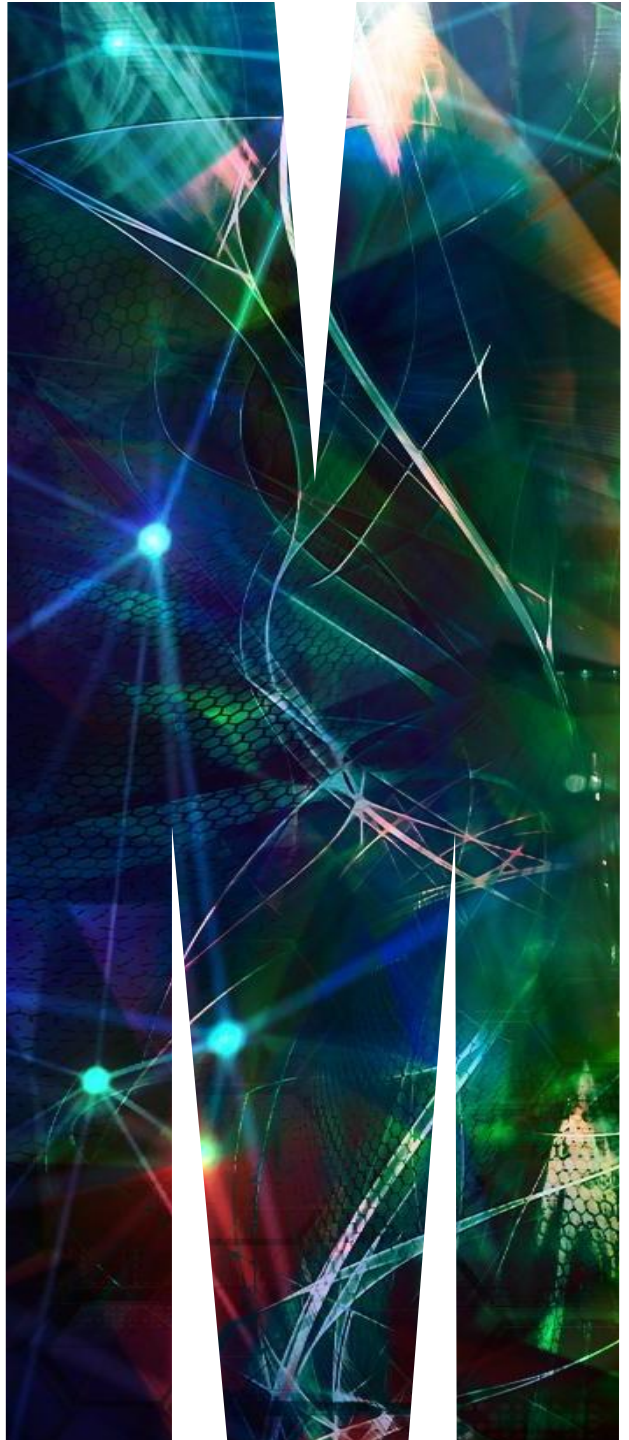
MONASH University

FIT2101- Software Engineering Process and Management

Guide to Project Management

Associate Professor Rashina Hoda (Clayton)
Dr Nisal de Silva (Clayton)
Ms Kamala Velautham (Malaysia)

Faculty of Information Technology



MONASH University

Project Plan

Project Plan

- Document that contains (or links to) the information your team needs to manage itself

- Usually an internal document
  - written for the team itself
  - not usually given to clients
  - contains policies to be followed by everyone
  - developed during project inception; controls the rest of the project
  - good document to hand to a new starter

- Used in both Agile and traditional projects

What's in a Project Plan?

- Info about the project
  - what it is, who it's for, what needs to be delivered and when
  - other systems it needs to work with
  - who to talk to in other organizations (e.g. client, vendors, etc.) and who's responsible for talking to them

- Info about the team
  - who's on it
  - roles, responsibilities, who's in charge of what (if applicable)
  - contact info

What's in a Project Plan? (cont'd)

- Info about the process
  - what process model is being used? (e.g. Scrum? Spiral? Unified? how long are iterations/sprints?)
  - policies for keeping everyone informed – meetings? written reports?
  - how will we allocate tasks to people? how will we keep track of the amount of time they are spending?

- Info about the context
  - what other stakeholders are there?
  - what is their interest in the project?
  - how much influence do they have?

Begin with a summary

- Brief description of project: what's it supposed to do? who is it for?
  - careful not to fall into adspeak – remember this document is to be read by your team
  - focus on why the project is being developed, not on how or on its design – those are covered in other sections

List the deliverables

- What are you planning to hand over, and when?
  - if your deadlines change, e.g. because you've renegotiated or because the requirements have changed, make sure you keep this up to date
  - should include both external
  - deliverables (for clients/end users) and internal deliverables (for your team, or other groups in your company)

Explain how the project is organized

- Technical term: project context

- What process model will you use? (e.g. Scrum, Unified Rational Process, Lean, Spiral)
  - are you going to be tailoring it to your own circumstances? if so, how?
  - why did you select this model?
  - give details, e.g. if you're using Scrum, how long will the sprints be?

- What tools and techniques are you going to use?
  - not just for programming, but for all other project activities
  - Analysis, design, testing, integration, etc.

Project context (cont'd)

- How will your team be organized?
  - who is responsible for what?

- What about other groups?
  - other teams in your company who you need to collaborate with
  - clients and end users
  - vendors and suppliers of the software and services you're using
  - need to know who they are, how to contact them, who's responsible for dealing with them

Establish your Vision

- First, you need to figure out what you're building
  - don't want to try to spec out all requirements
  - but do need to know the big picture

- You will be making decisions about requirements later on
  - "should we implement feature, or is it out of scope?"
- Can't make these decisions without having some idea of what the scope is.

- One possible approach: write a brief vision statement or elevator pitch for your product
  - "elevator pitch": something you'd be able to say to an interested venture capitalist/potential client if you bumped into them in an elevator
  - for our purposes, vision statements and elevator pitches are interchangeable
  - they are both used to clarify your project's goals

Vision statements/elevator pitches

- Not the same as a mission statement; those are for the company as a whole
- Keep it brief: ideally, one or two sentences.
- Should be understandable by all stakeholders in the project
    - including senior management and clients, not just the team
- Be precise
    - We will delight our customers…
    - …by doing what?
- Be specific
    - We expect to make a profit
    - this does not differentiate your project from any other project in history
- …But don't be too specific
    - We will use Agile practices to implement a website using HTML5 and React.js over a MySQL back end…
    - apart from being verbose, you might want to change this later

Elevator pitch template

- From Jonathan Rasmusson, The Agile Samurai:

> For [*target customer*]
> who [*statement of need or opportunity*]
> the [*product name*]
> is a [*product category*]
> that [*key benefit, compelling reason to buy*].
> Unlike [*primary competitive alternative*]
> our product [*statement of primary differentiation*].

Elevator pitch example

- From Jonathan Rasmusson, The Agile Samurai:

For *construction managers*
who *need to track what type of work is being done on the construction site*
the *CSWP*
is a *safety work permit system,*
that *creates, tracks and audits safety work permits*.
Unlike *the current paper-based system*
our product *is web based and can be accessed any time from anywhere*.

Roles, responsibilities, and tasks

- Your role is essentially a job title
  - it's possible to perform multiple roles on the same project
  - common on cross-functional teams
- Your responsibilities are the things you're supposed to do, or ensure are done
- Tasks are things that need to be done
- The Project Plan needs to define the role and responsibilities of each team member
  - it *doesn't* need to specify which tasks each team member must do
  - it *does* need to say how tasks will be allocated and tracked

A word about team structure

- Common problem with student teams: lack of structure
  - “we will make decisions by consensus”
- Don’t want to boss other students around
- Don’t want too much to do
- But… don’t have a Scrum Master present to help achieve consensus
  - so this approach doesn’t always work

- Problem: *if everybody’s responsible, then nobody’s responsible*

- Don’t think of it as control, think of it as stepping up and taking responsibility
- Responsibility doesn’t mean you have to do it yourself, just that you have to make sure it’s done

Time and task management

- Time management
  - how will you keep track of what the developers are spending time on?
  - do you have standard working hours? Maximum/minimum commitments?

- Task management
  - how will you decide who does what?
  - how will you keep track of who is working on what?
    - for Scrum: where will your project/sprint backlogs live?

Why track time and tasks

- Make sure everything gets done
- Get a better idea of how long things take you
    - help you improve at time estimation, which is known to be hard
- Get some practice for industry
    - even salaried software engineers often have to fill in timesheets to keep track of which projects they've worked on

- Resolve disputes
    - lets you prove that each team member is pulling their weight
    - gives everyone an incentive to do their fair share
    - in FIT2101, your markers can use your time logs if there's any question about fairness of workload

MONASH University

Analysis of Alternatives

Architectural decisions

- Once you've decided what you're building, you need to make some important decisions about how you'll do it
- It's not time to do a detailed architectural design yet; you're only just beginning to think about requirements
    - class-level/module-level details will change as you refactor anyway

- Still need to make some high-level decisions
    - what language/s will we use?
    - what tools? will we need a database?
    - what is our platform – mobile, desktop, web application?
- These decisions are very expensive to change during development
    - so need to make them with care

Justifying your decisions

- Should write down your decisions and the reasoning behind them
  - make sure everyone on the team understands them
  - justify any purchasing to senior management
  - if they do need to change later, you can prove to your client or manager that you did make a good decision based on what you knew at the time

- Can use a kind of report called an analysis of alternatives
  - not usually part of your Project Plan
  - for your project, you'll create one at the same time

Analysis of alternatives - structure

- One-page summary (if long)
- Glossary of terms (if needed)
  - or reference/hyperlink to team wiki, Google Doc, etc.
- Terms of reference
  - what factors are you taking into consideration?
  - how important is each one?
  - which options are you considering?
- Body
  - how does each option do against each of the factors?
  - did your investigations uncover anything else?
  - you might want to summarize this in a table
- Recommendations
  - justify your choices by reference to the findings presented in the body

Analysis of alternatives - Terms of reference

- What decision are you trying to support?
  - example: *we want to choose an IDE for all team members to use*
- What options have you identified?
  - example: *Atom, PyCharm, PyDev, Visual Studio*
- What are the criteria on which you will base your recommendations?
  - e.g. if considering which IDE to use, you might consider:
    - *cost*
    - *supported languages (must support Python; JavaScript/HTML nice to have)*
    - *licence must allow commercial development*
    - *Speed*
  - in the body, you evaluate each option against each of these criteria
-
-

Analysis of alternatives - Body

- The body is where you present your evidence for and against each option
- It should list each option systematically and compare relevant features
    - *relevant* means *in the terms of reference*
    - if it's not in the terms of reference, that means you don't care
- Keep the body brief, factual, and to the point
    - if you're offering an opinion rather than fact, make it clear that you are doing so
    - If you've got facts, back them up with sources
- Note that you might need to expand your list of options as you research
    - e.g. PyCharm and Visual Studio both have Professional and Community Editions

Analysis of alternatives - Recommendations

- Here, you tie it all together
  - the terms of reference say what you consider important
  - the body goes through the options and evaluates one each against the criteria established in the terms of reference
  - in the recommendations section, you make it clear which option/s will satisfy the criteria the best
  - sometimes there will be a clear winner, other times there won't
    - if there's no clear winner, you need to help your readers decide

MONASH University

PO management

External Product Owner Management

- Your clients/POs will be wandering around in the workshops.
  - Ask questions if you are unclear about the requirements given so far.
  - No assumptions for requirements!
- Additionally, you can meet them in PO consultations.

- A good way to finalize the requirements is to put them into a shared document in your shared Google Drive folder.

- You can include the following (but not limited to) in this document.
  - What information would your client like your program to show them?
  - How would they like to be able to access it?
  - What options will they have over how this information is presented, and how should they set these options?

External Product Owner Management (cont'd)

- Remember, this document is going to help you with,
  - Vision statement
  - Most appropriate approach to use during analysis of alternatives
  - Begin creating the product backlog

Happy Project Planning!

MONASH University