



THE UNIVERSITY OF
MELBOURNE

SWEN90016
Software Processes & Project Management

Project Scheduling

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

2021 – Semester 2

Lecture 5

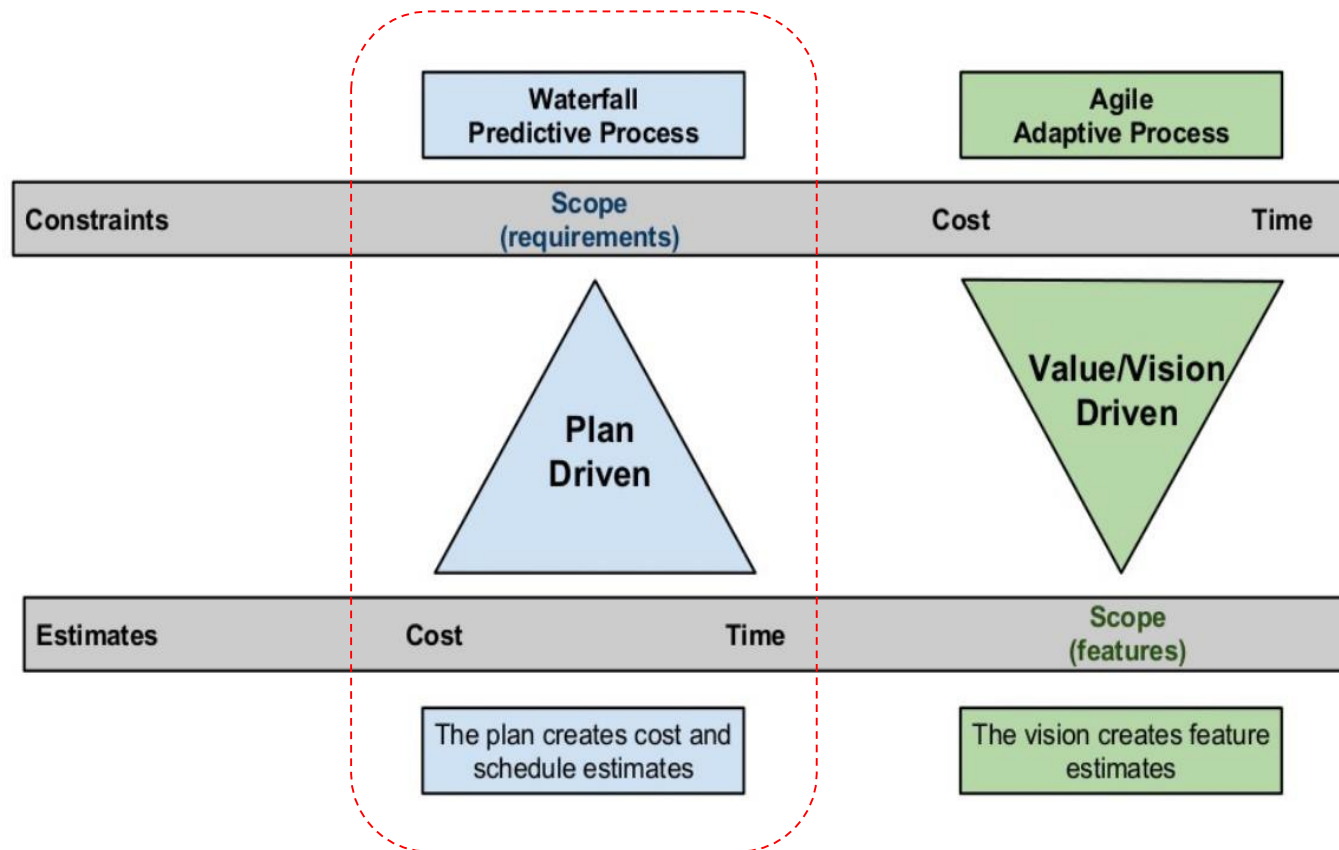
Copyright University of Melbourne 2018



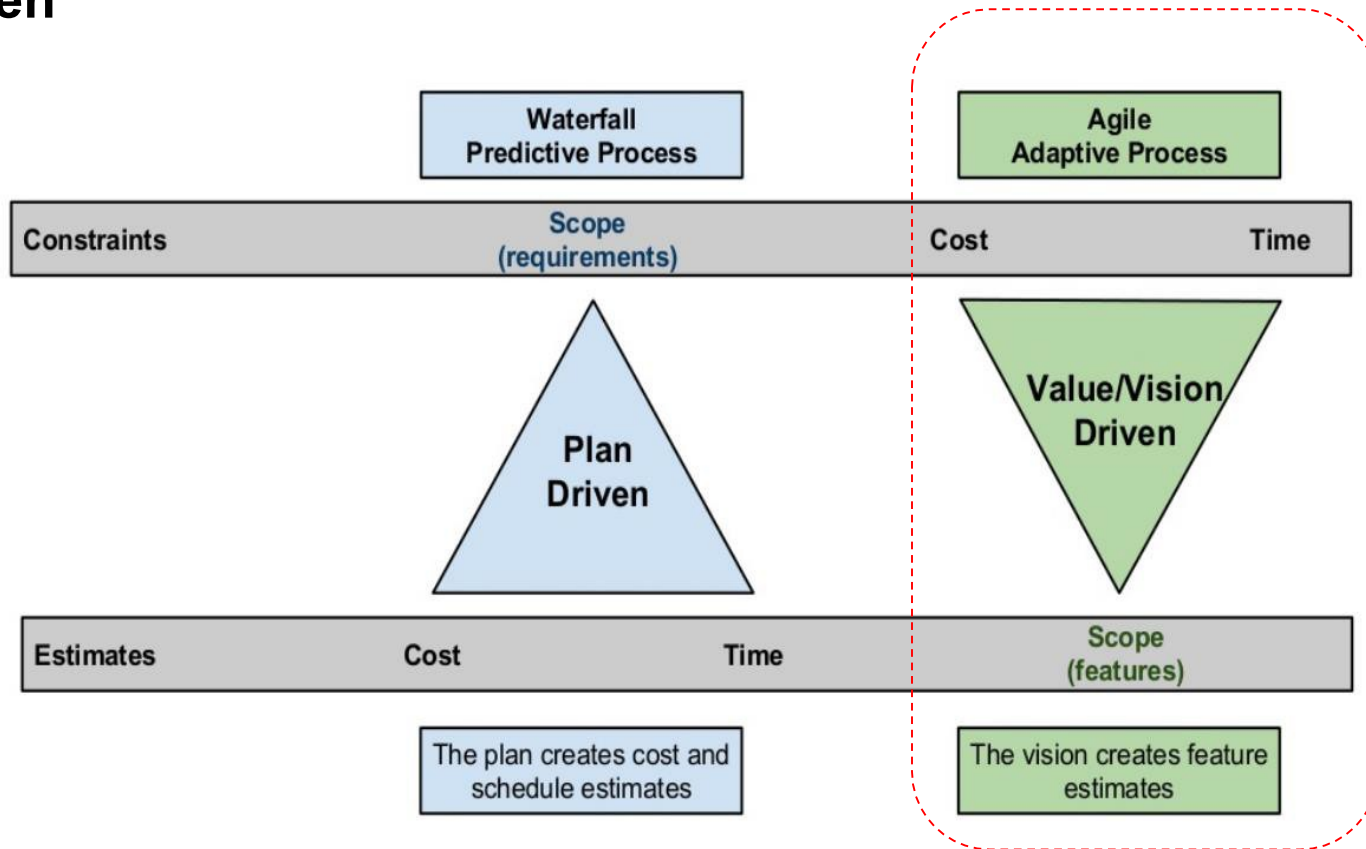
1. Understand the role of a project schedule
2. Understand how to develop a project schedule
3. Understand how to use a project schedule to monitor and track project progress
4. Understand agile planning principles

- **Project Schedule:**
 - One of the important artefacts generated during the project planning phase
 - Is used and maintained throughout the project to monitor and track project progress - is a living document
- **What does the project schedule contain?**
 - Duration and dependencies for each task
 - People and physical resources required by each task
 - Milestones and deliverables
 - Project Timeline

Project planning and scheduling introduced in this topic apply to formal SDLC processes – Plan Driven



Agile SDLC processes do not use a project schedule - Value/Vision Driven



Anecdotally organizations that use Agile practices also use project schedules for budgeting, contracting and reporting purposes.



1. Understand the role of a project schedule
2. Understand how to develop a project schedule
3. Understand how to use a project schedule to monitor and track project progress
4. Understand agile planning principles



1. Breakdown the task into small chunks you can deal with – **Work Breakdown Structure (WBS)**
2. Identify the **interdependencies** between the broken down tasks and develop a **task network**
3. Estimate the **effort** and the **time allocation** for each task
4. **Allocate resources** for tasks and validate effort
5. Develop the **project schedule**



- Planning and executing large tasks is challenging:
 - Estimating the time and resources
 - Identifying interim goals and deliverable
 - Progress monitoring
- Solution is to break the task down to manageable units:
 - Each task should have a specific outcome or a deliverable
 - Results in a Work Breakdown Structure (WBS)

Example - WBS

Redecorate Room

Prepare materials

- Buy paint
- Buy a ladder
- Buy brushes/rollers
- Buy wallpaper remover

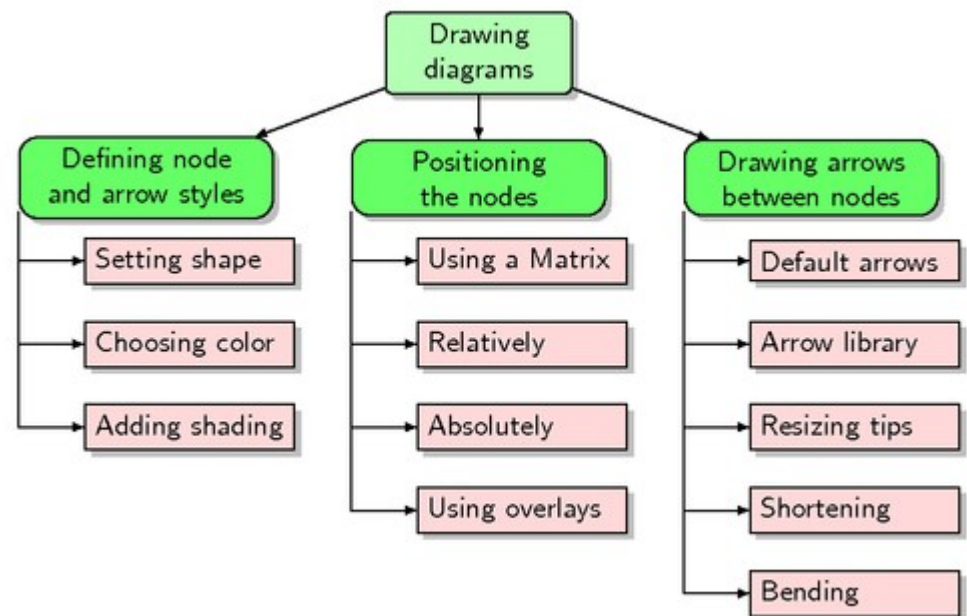
Prepare room

- Remove old wallpaper
- Remove detachable decorations
- Cover floor with old newspapers
- Cover electrical outlets/switches with tape
- Cover furniture with sheets

Paint the room

Clean up the room

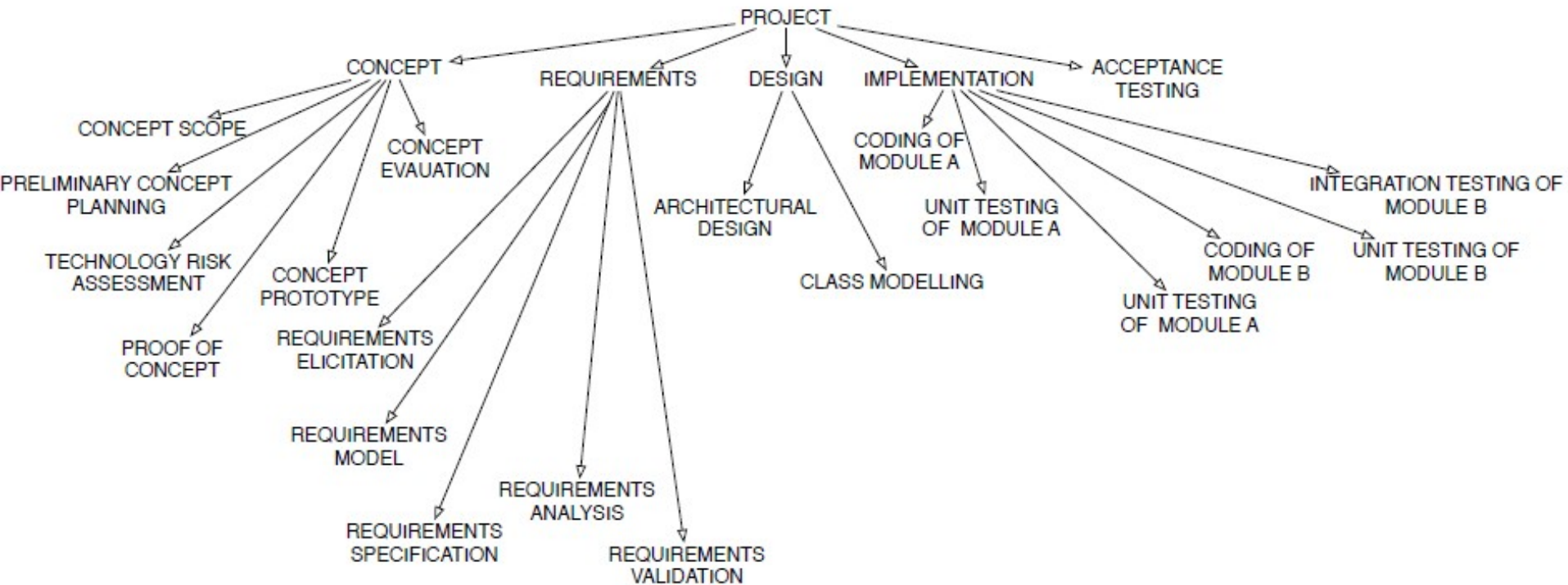
- Dispose or store leftover paint
- Clean brushes/rollers
- Dispose of old newspapers
- Remove covers



<http://texample.net/tikz/examples/work-breakdown-structure/>

<http://slideplayer.com/slide/5384158/>

Example – WBS (Software Project)





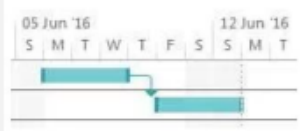


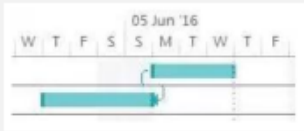
Developing a Project Plan

MELBOURNE

1. Breakdown the task into small chunks you can deal with – Work Breakdown Structure (WBS)
2. Identify the **interdependencies** between the broken down tasks and develop a **task network**
3. Estimate the effort and the time allocation for each task
4. Allocate resources for tasks and validate effort
5. Develop a project schedule

- **Tasks can be:**
 - **Unconstrained:** the task can start at any time (buy paint, remove detachable decorations)
 - **Constrained:** depends on another task (cannot remove wall paper until decorations are removed)
 - If task **B** depends on task **A** (**A ->B**)
 - **B** is a Successor task (S)
 - **A** is a Predecessor task (P)
 - Remove Detachable Decorations (P) -> Remove wall paper (S)
- **Dependencies are caused by:**
 - a task needing a work product of another task
 - a task needing resources used by another task

Types of Task Dependencies

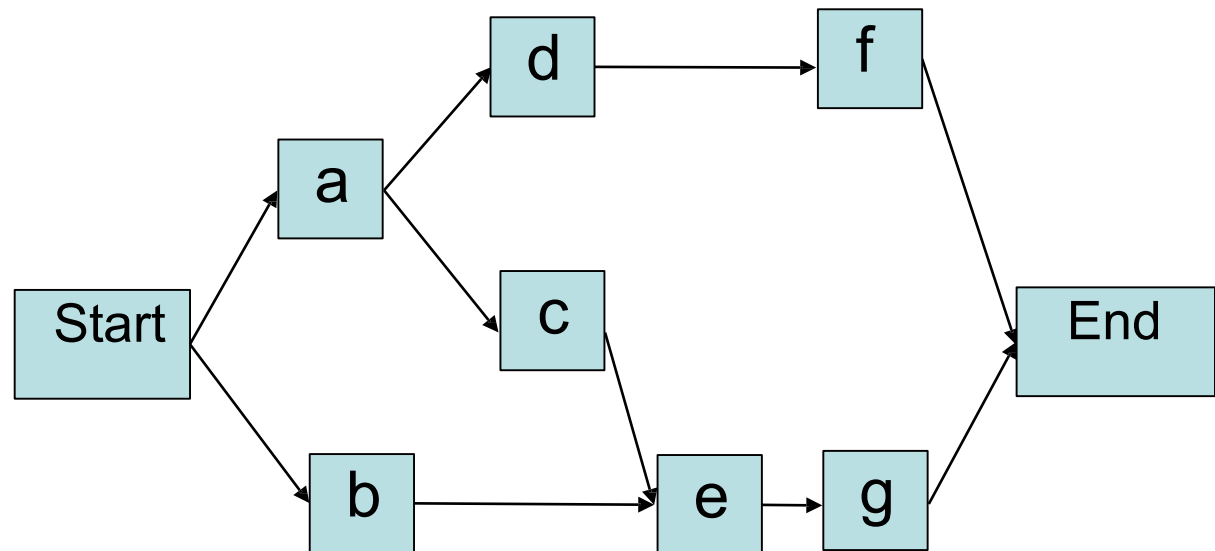
Dependency	Description	Representation
Finish-to-Start	Predecessor must finish before Successor can start	
Start-to-Start	Predecessor must start before Successor can start	
Finish-to-Finish	Predecessor must finish before the Successor can Finish	
Start-to-Finish	Predecessor must start before the Successor can finish	

The most common type of dependency is the finish-to-start dependency

Task Network

MELBOURNE

Activity	Predecessor
<i>a</i>	—
<i>b</i>	—
<i>c</i>	<i>a</i>
<i>d</i>	<i>a</i>
<i>e</i>	<i>b, c</i>
<i>f</i>	<i>d</i>
<i>g</i>	<i>e</i>

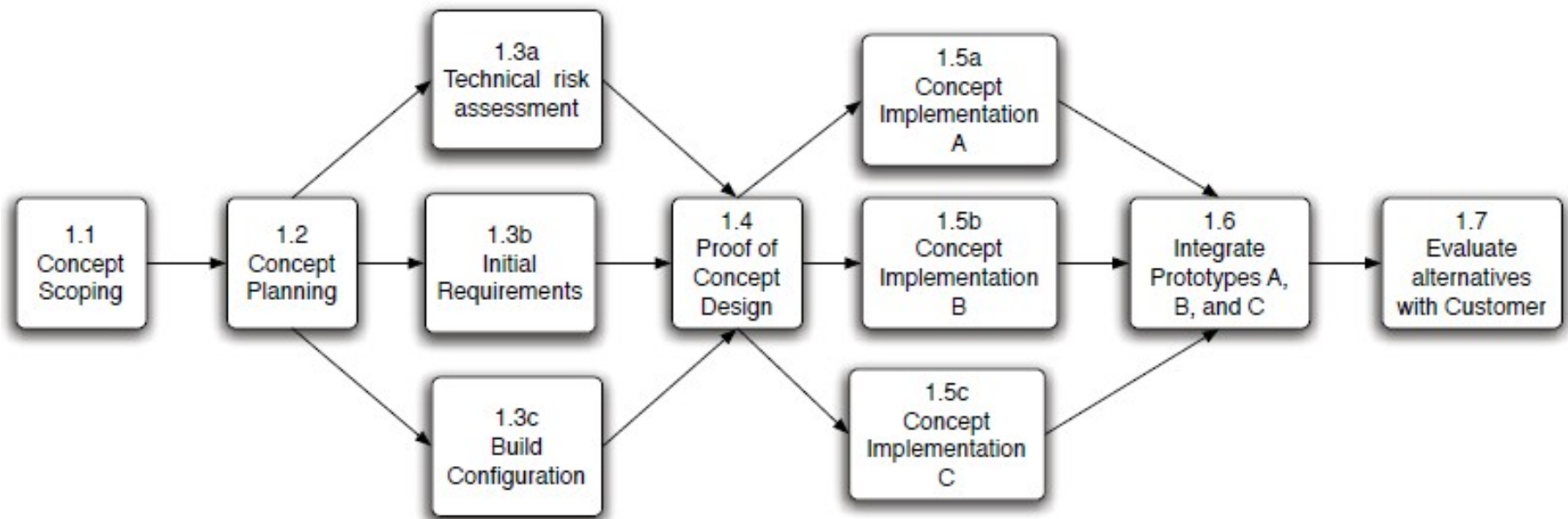


WBS – Software Project

MELBOURNE

1. Concept
 1. Concept Scope
 2. Preliminary Concept Planning
 3. Preliminary Analysis
 - 1.3a Technology Risk Assessment
 - 1.3b Initial Requirements
 3. c Build Configuration
 4. Proof of Concept
 5. Concept Prototype
 6. Prototype Integration
 7. Concept Evaluation
2. Requirements
 1. Requirements Elicitation
 2. Requirements Prototype
 3. Requirements Analysis
 4. Requirements Specification
 5. Requirements Validation
3. Design
 1. Software Architecture Design
 2. Class Models
4. Implementation
 1. Coding the Client
 2. Testing the Client
 3. Coding the Server
 4. Testing the Server
 5. Integration Testing of Client with Server
5. Acceptance Testing

Task Network – Software Project





Developing a Project Plan

MELBOURNE

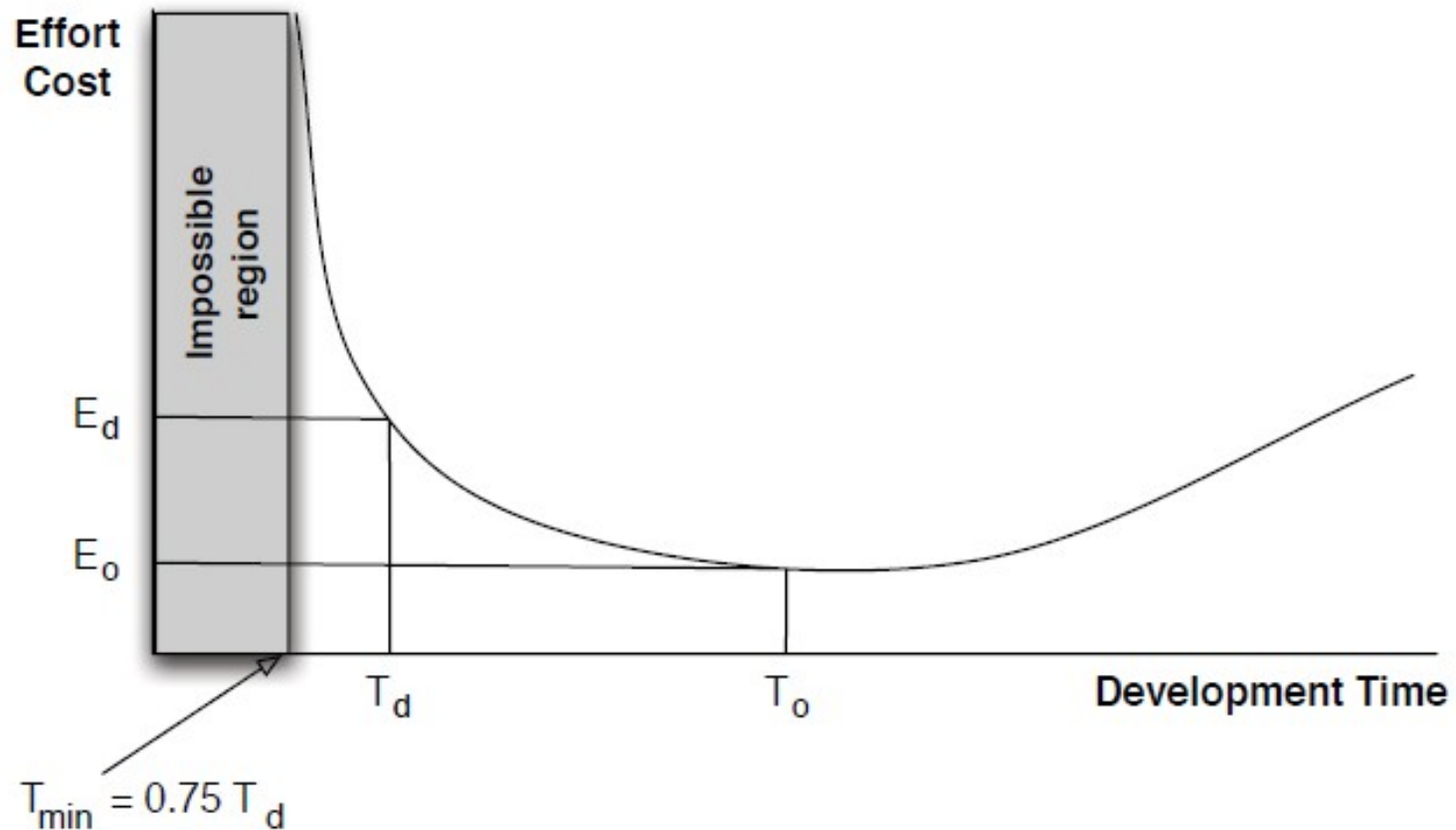
1. Breakdown the task into small chunks you can deal with – Work Breakdown Structure (WBS)
2. Identify the interdependencies between the broken down tasks and develop a task network
3. Estimate the **effort** and the **time allocation** for each task
4. Allocate resources for tasks and validate effort
5. Develop a project schedule



MELBOURNE

- **A common measure for estimating the effort for software is *man-months* (more generally *person-months*)**
 - Effort estimation will be covered in week 7
- **person-months:**
 - the time in months for a single person working full time to complete the task
- **The Mythical Man-Months [Brooks seminal paper]**
 - man-months is a misleading measure to estimate software
 - adding people to a project that is behind schedule could result in more damage than helping it

Effort vs Time



Putnam-Norden-Rayleigh curve



- **Terminology**

optimistic time - O

pessimistic time - P

most likely time - M

expected time - T_E

$$T_E = (O + 4M + P)/6$$

Time Estimation

MELBOURNE

Activity	Predecessor	Time estimates			Expected time (T_E)
		Opt. (O)	Normal (M)	Pess. (P)	
a	—	2	4	6	4.00
b	—	3	5	9	5.33
c	<i>a</i>	4	5	7	5.17
d	<i>a</i>	4	6	10	6.33
e	<i>b, c</i>	4	5	7	5.17
f	<i>d</i>	3	4	8	4.50
g	<i>e</i>	3	5	8	5.17



Developing a Project Plan

MELBOURNE

1. Breakdown the task into small chunks you can deal with – Work Breakdown Structure (WBS)
2. Identify the interdependencies between the broken down tasks and develop a task network
3. Estimate the effort and the time allocation for each task
4. **Allocate resources** for tasks and validate effort
5. Develop a project schedule

- If the effort (person-months) and the time are known, the number of personnel can be computed as:

$$N = \frac{\text{Effort}}{T}$$

- Assigning people to tasks
 - Although computing the number of personnel required for each task appears simple, resource allocation is complicated task
 - The project manager has to carefully consider the expertise of the people, and the availability of them for tasks, which might require validation and adjustment of the schedule



Developing a Project Plan

MELBOURNE

1. Breakdown the task into small chunks you can deal with – Work Breakdown Structure (WBS)
2. Identify the interdependencies between the broken down tasks and develop a task network
3. Estimate the effort and the time allocation for each task
4. Allocate resources for tasks and validate effort
5. Develop a **project schedule**



Developing a Project Plan

MELBOURNE

1. Breakdown the task into small chunks you can deal with – Work Breakdown Structure (WBS)
2. Identify the interdependencies between the broken down tasks and develop a task network
3. Estimate the effort and the time allocation for each task
4. Allocate resources for tasks and validate effort
5. Develop a **project schedule**

MELBOURNE

- **Project Schedule will answer two important questions not answered so far:**
 - How long will the system take to develop?
 - How much will it cost?
- **Two widely used graphical notations to represent the Project Schedule**
 - Gantt charts
 - A bar chart that shows the schedule against a calendar
 - PERT (Program Evaluation and Review Technique) charts
 - An activity network that shows the dependencies among tasks and the *critical path*

Project Scheduling - Definitions

MELBOURNE

Term	Description
Activity (Task)	Is part of a project that requires resources and time
Milestone	Is the completion of an activity that provides evidence of a deliverable completion or end of a phase – is an event that takes zero time
Free float (free slack)	Is the amount of time that a task can be delayed without causing a delay to subsequent tasks
Total float (total slack)	Is the amount of time that a task can be delayed without delaying project completion
Critical path	Is the longest possible continuous path taken from the initial event to the terminal event
Critical activity	Is an activity that has total float equal to zero

- **Milestones**

- Mark specific points along a project timeline
- These points may signal anchors such as:
 - a project start and end date
 - a need for external review
 - start and end of a phase
 - a completion of a deliverable

- **Deliverable**

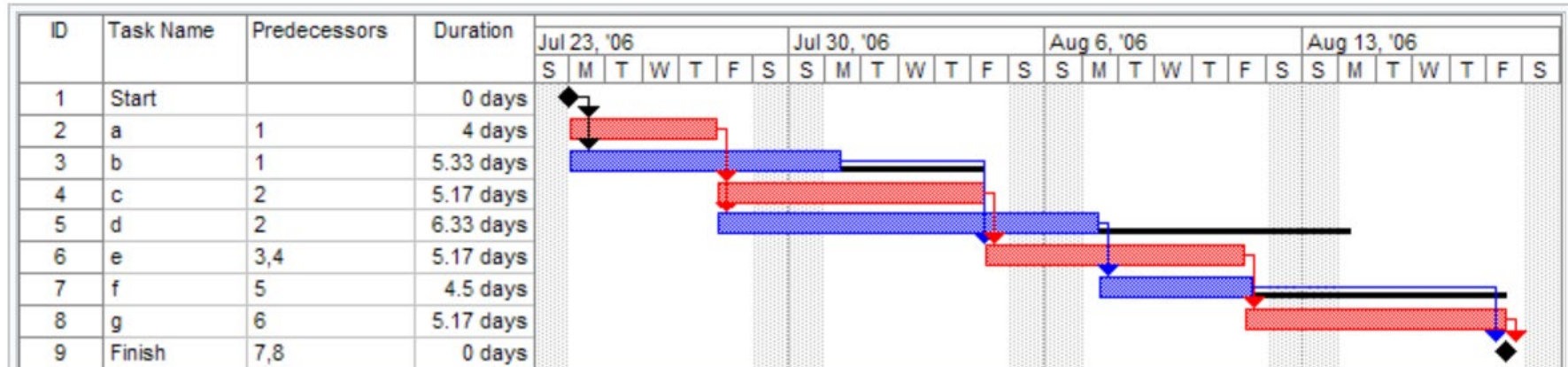
- Specific artefacts that are of interest
- Examples of deliverables include:
 - Project documents such as the Project Management Plan, Requirements Specification, Design Document, Test Plan etc.
 - Prototypes
 - Final application



- Was introduced by Henry Gantt in 1910
- Gantt chart is a horizontal bar chart which shows tasks against a timeline – **project schedule**
- Can be used to view planned activities vs progress and therefore is a useful tool for monitoring project progress

Gantt Chart

MELBOURNE



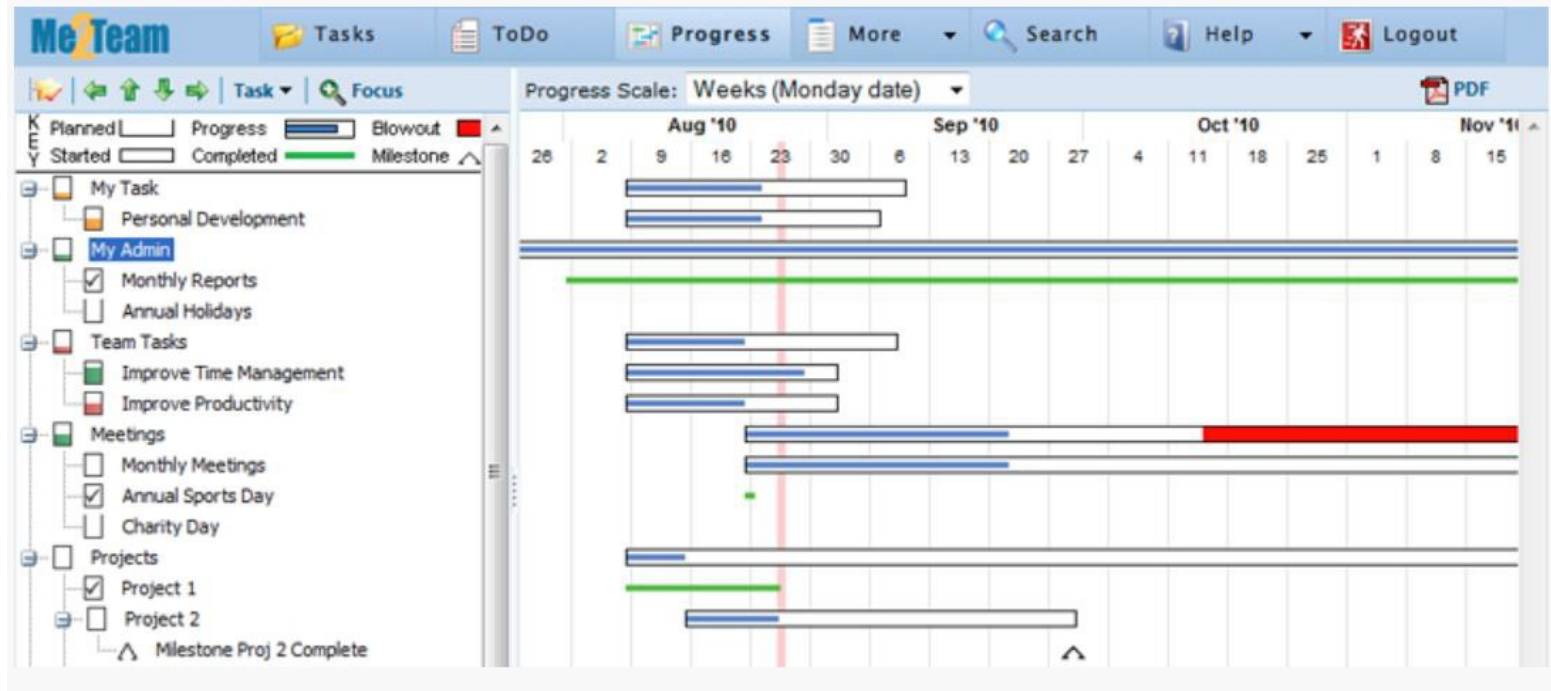
A Gantt chart created using [Microsoft Project \(MSP\)](#). Note (1) the **critical path** is in red, (2) the **slack** is the black lines connected to non-critical activities, (3) since Saturday and Sunday are not work days and are thus excluded from the schedule, some bars on the Gantt chart are longer if they cut through a weekend.

Linked Gantt charts

- contain lines indicating the dependencies between tasks

Gantt Chart

MELBOURNE



Progress Gantt charts

- tasks are shaded in proportion to the degree of their completion
- used for progress tracking – gives a visual representation of the progress

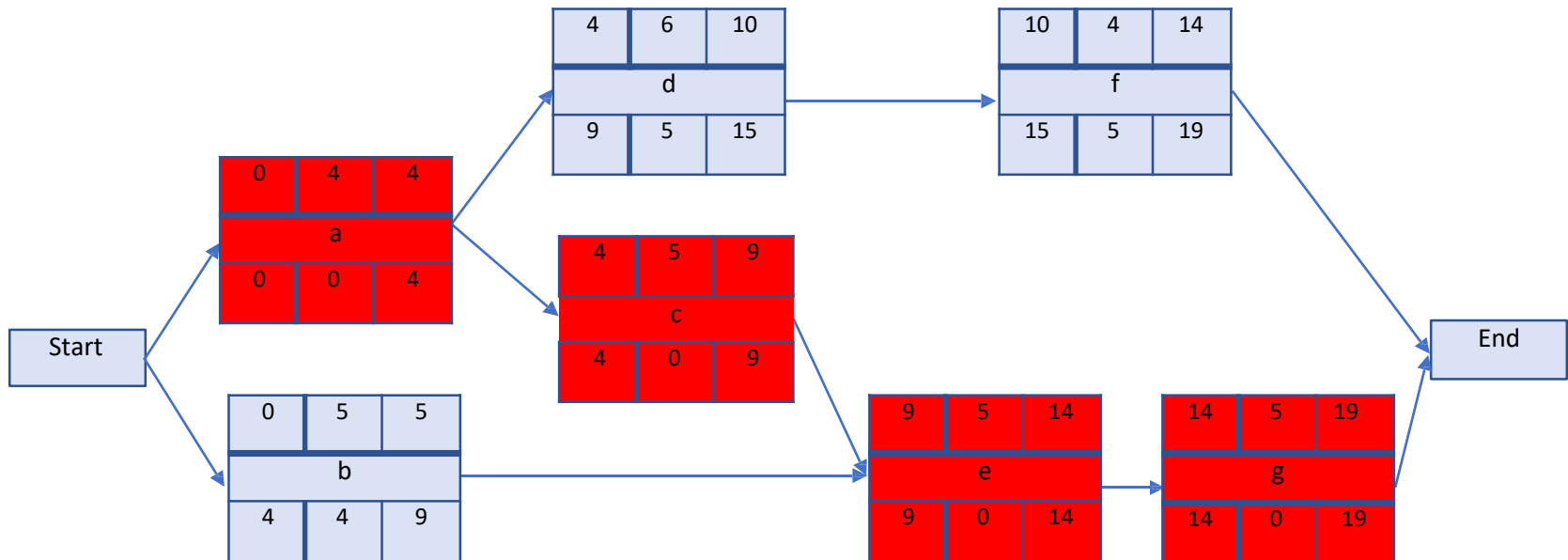


- PERT (Program Evaluation and Review Technique) chart:
 - A task network which shows the dependencies along with time related information and the critical path
- PERT analysis helps:
 - understand the characteristics of the project that will let project managers do scheduling trade-offs
 - perform critical path analysis
 - monitor project progress and re-plan

- Involves calculating the following estimates:
 - Earliest start time (ES)
 - Latest start time (LS)
 - Earliest finish time (EF)
 - Latest finish time (LF)
 - Slack time

ES	Duration	EF
Task Name		
LS	Slack	LF

MELBOURNE



Critical Path: a, c, e, g
 Duration: 19 days

Notes:

- Critical path activities have a total free slack of 0
- Two parallel paths could be critical paths
- There can be more than one critical path



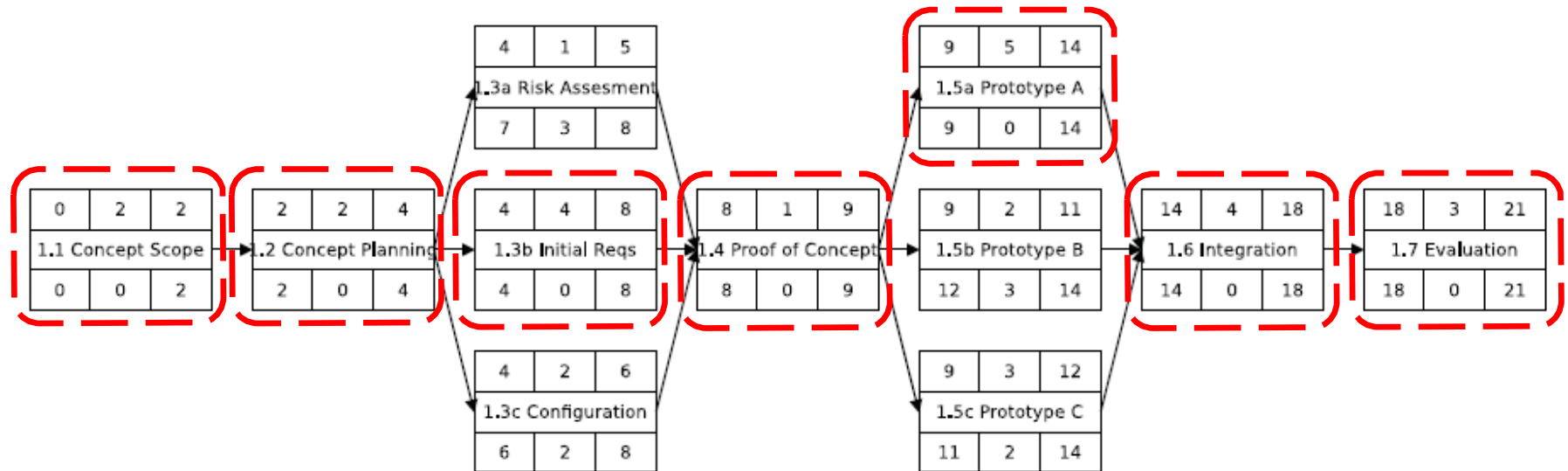
PERT Chart Example

MELBOURNE

Task	Dependencies	Most Likely Time
1.1 Concept Scoping		2 days
1.2 Concept Planning	1.1	2 days
1.3a Technology Risk Assessment	1.2	1 day
1.3b Initial Requirements	1.2	4 days
1.3c Configuration	1.2	2 days
1.4 Proof of Concept	1.3a, 1.3b, 1.3c	1 day
1.5a Concept Prototype A	1.4	5 days
1.5a Concept Prototype B	1.4	2 days
1.5a Concept Prototype B	1.4	3 days
1.6 Prototype Integration	1.5a, 1.5b, 1.5c	4 days
1.7 Concept Evaluation	1.6	3 days

PERT Chart Example

MELBOURNE



Critical Path: 1.1, 1.2, 1.3b, 1.4, 1.5a, 1.6, 1.7

Duration: 21 days

Note: Critical path activities have a total free slack of 0











MELBOURNE

- **Critical Path**
 - path with the longest duration
 - activities on the critical path have a total free slack of 0
 - a delay in any of the activities in the critical path will cause the project to delay
- **Crashing the project schedule**
 - shortening the total duration of the project by shortening the critical path
 - By removing the dependencies between activities in the critical path; or
 - Shortening the duration of activities in the critical path

Tools (for reference)

MELBOURNE

Atlassian (Jira), Microsoft project

Product	Rating	Price	Platforms	Deployments	Business Size	
 Smartsheet	★★★★☆ (395)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Mavenlink	★★★★☆ (224)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Workzone	★★★★☆ (38)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 inMotion	★★★★☆ (32)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Accelo	★★★★☆ (3)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 monday.com (formerly dapulse)	★★★★☆ (606)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Workfront	★★★★☆ (425)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Freshservice	★★★★☆ (341)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Wrike	★★★★☆ (745)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website
 Airtable	★★★★☆ (162)	\$\$\$\$\$	🍏 🖥️ 📱	☁️ 📧	S M L	Visit Website

<https://www.workzone.com/blog/gantt-chart-software/>



- Understand the role the project schedule
- Understand how to develop a project schedule
- Understand how to use a project schedule to monitor and track project progress
- Understand agile planning principles

- **How do software projects fall behind schedule?**

One day at a time

– *Fred Brooks, the well-known author of the seminal article **Mythical Man-Months***

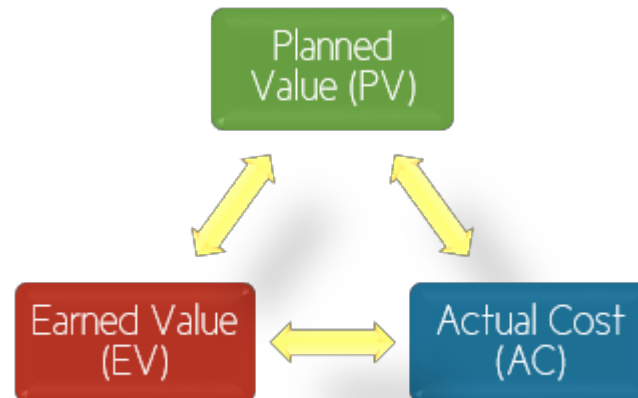
- **Project scheduling is important, but *tracking and controlling* are even more important!**

- How to track and control project progress?
 - Periodic meetings where team members report progress
 - Evaluating the results of reviews and audits conducted as part of the software engineering process
 - Tracking formal project milestones
 - Comparing actual start dates with scheduled start dates
 - Meeting engineers and having informal discussions
 - Using a formal method like *earned value analysis*

Earned Value Analysis (EVA)

MELBOURNE

- EVA can be used to:
 - report current/past project performance
 - predict future project performance based on current/past performance
- Results can be expressed in dollars and/or percentage



- Planned Value (PV)
 - that portion of the approved cost estimate planned to be spent on the given activity during a given period
- The Earned Value (EV)
 - the value of the work actually completed
- Actual Cost (AC)
 - the total of the costs incurred in accomplishing work on the activity in a given period

- Consider the following scenario:

You are assigned to manage a project that is planned to finish in 12 months, estimated to cost \$100,000. At the end of the third month, based on the project Gantt chart, 20% of the work had been reported as completed. The finance department has reported the cost of the project to date as \$35,000.

What is the PV?

What is the EV?

What is the AC?

- Consider the following scenario:

You are assigned to manage a project that is planned to finish in 12 months, estimated to cost \$100,000. At the end of the third month, based on the project Gantt chart, 20% of the work had been reported as completed. The finance department has reported the cost of the project to date as \$35,000.

$PV = \$100,000 * 3/12 = \$25,000$ (assuming equal work distribution over the period, which may not be the case always)

$EV = \$100,000 * 20/100 = \$20,000$

$AC = \$35,000$

- **Schedule Variance Analysis**
 - Uses EV and PV to calculate a variance to the project schedule

- **Schedule Variance: expressed in dollars**

$$\begin{aligned}SV &= EV - PV \\&= 20,000 - 25,000 \\&= (5000)\end{aligned}$$

- **Schedule Performance Index: expressed as a fraction**

$$\begin{aligned}SPI &= EV/PV \\&= 20,000/25,000 \\&= 0.8\end{aligned}$$

- **Cost Variance Analysis**
 - Uses EV and AC to calculate a variance to the project schedule

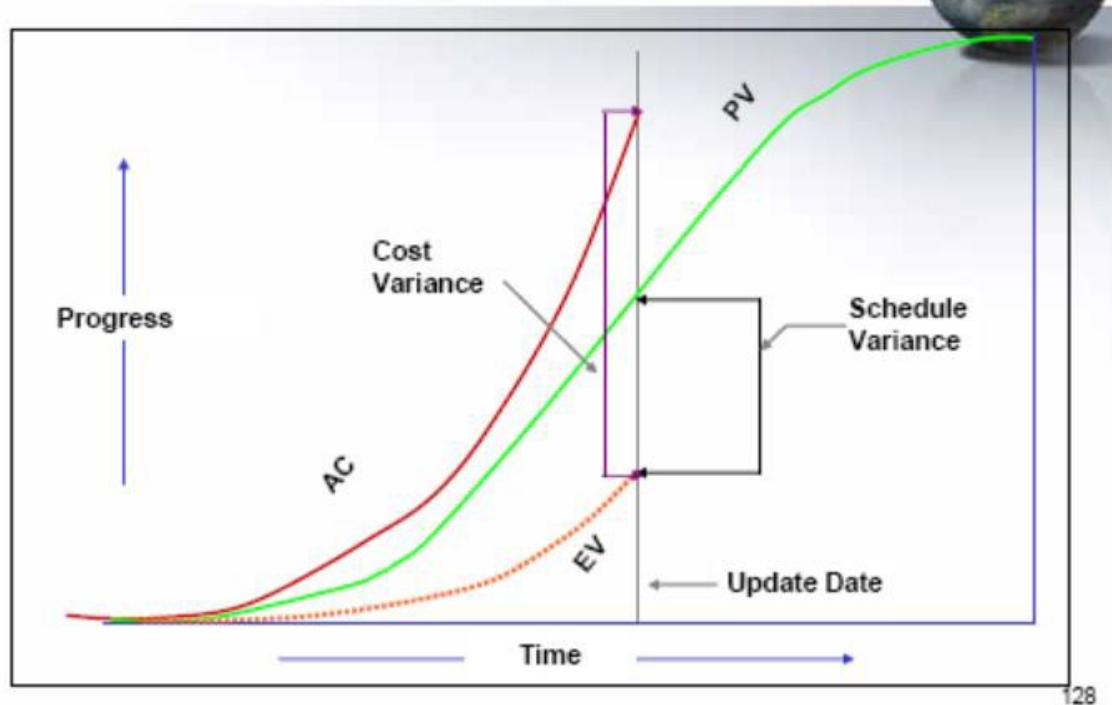
- **Cost Variance: expressed in dollars**

$$\begin{aligned} CV &= EV - AC \\ &= 20,000 - 35,000 \\ &= (15,000) \end{aligned}$$

- **Cost Performance Index: expressed as a fraction**

$$\begin{aligned} CPI &= EV/AC \\ &= 20,000/35,000 \\ &= 0.57 \end{aligned}$$

Graphic Performance Report



<https://www.pmi.org/learning/library/earned-value-management-systems-analysis-8026>



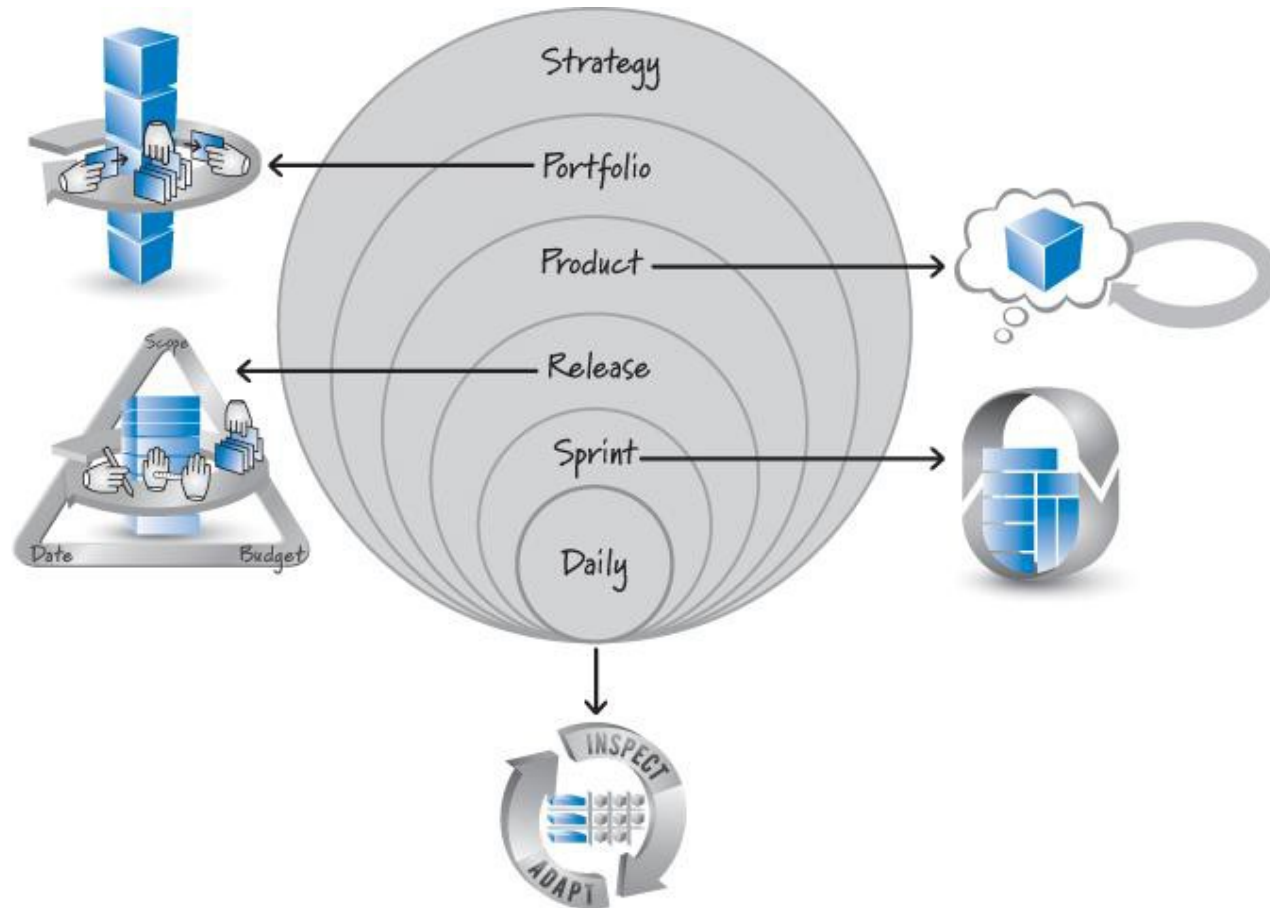
1. Understand the role of a project schedule
2. Understand how to develop a project schedule
3. Understand how to use a project schedule to monitor and track project progress
4. Understand agile planning principles

- Takes a significantly different flavour from traditional approaches
- Detailed planning is deferred until the start of the iteration
 - Designed to handle change
 - An iteration includes all phases (requirements, design and test)
- Planning is based on light weight lists
 - Gantt and PERT charts are considered less useful



- Plan short iterations
- Deliver working software
- Use “Just in time (JIT) planning” – next iteration
- Use the team

Planning in Scrum

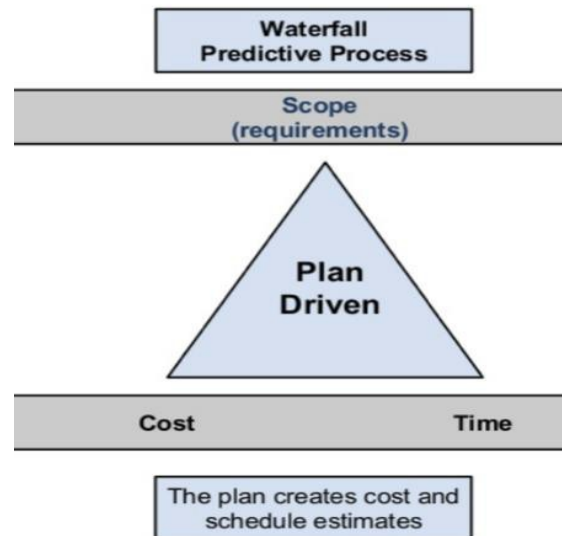


Different levels of planning in Scrum

Planning in Scrum

Level	Horizon	Who	Focus	Deliverables
Portfolio	Possibly a year of more	Stakeholders and product owners	Managing a portfolio of products	Portfolio backlog and collection of in-process products
Product (envisioning)	Up to many months or longer	Product owner, stakeholders	Visions and product evolution over time	Product vision, roadmap, and high-level features
Release	Three (or fewer) to nine months	Entire Scrum Team, Stakeholders	Continuously balance customer value and overall quality against the constraints of scope, schedule and budget	Release Plan
Sprint	Every iteration (one week to one month)	Entire Scrum Team	What features to deliver in the next Sprint	Sprint goals and sprint backlog
Daily	Every day	Scrum Master, development team	How to complete committed features	Inspection of current progress and adaptation

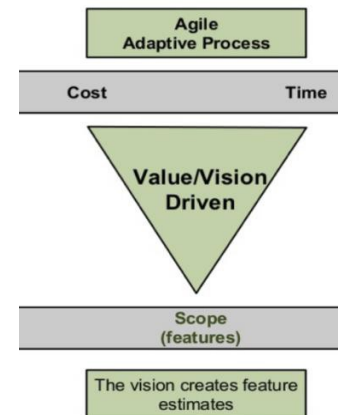
- Assumptions in Formal Planning:
 - Scope fixed – requirements are stable
 - Budget fixed – cost estimations are accurate
 - Schedule fixed - derived based on scope and budget



Release Planning

- Agile Planning

- Recognizes that all three factors: scope, budget and time cannot be fixed in reality - not recommended
- Can we fix scope and date and make the budget flexible?
 - Not really because increasing the budget, hence the resources will not always help to improve speed – not recommended
- So what are our options?
 - Fix date and budget
and have the scope flexible
Fixed-Date release planning



- Fix scope and have the date and budget flexible – *Fixed-Scope release planning*

e.g. <https://www.aoe.com/en/company/agile-teams.html>

Fixed-Date Release Planning

MELBOURNE

Determine the number of sprints N
 $N = \text{total duration} / \text{length of sprint}$

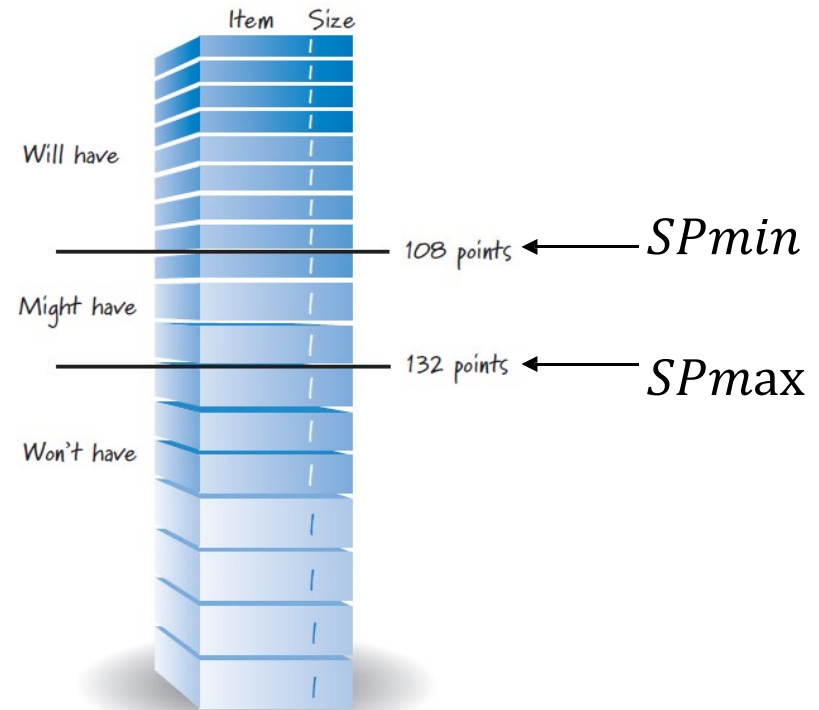
Groom the product backlog by estimating and prioritizing stories

Measure team velocity range:
 V_{min}, V_{max}

Compute minimum and maximum story points based on velocity
 $SP_{min} = V_{min} \times N,$
 $SP_{max} = V_{max} \times N$

Draw lines through the Product Backlog to show the above

Fixed-Date: used when date is more important



Fixed-Scope Release Planning

MELBOURNE

Groom the product backlog by creating, estimating and prioritizing and identify the must-have stories

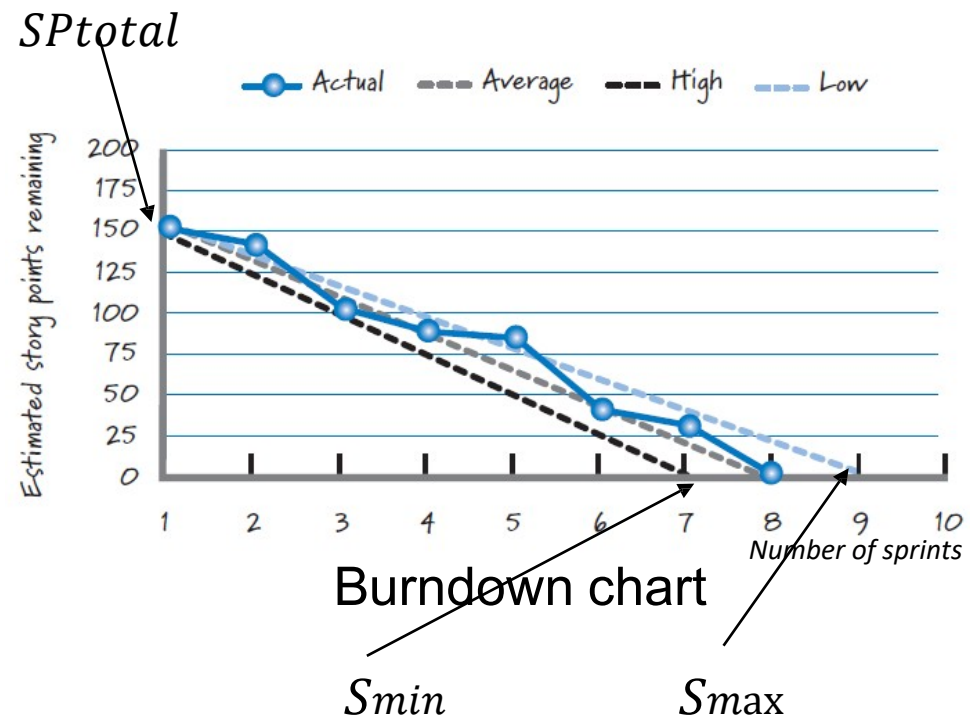
Determine the total number of must-have story points (SP_{total})

Measure team velocity range:
 V_{min} , V_{max}

Compute minimum and maximum number of sprints
 $S_{min} = SP_{total} / V_{max}$,
 $S_{max} = SP_{total} / V_{min}$

Show on Burndown Chart

Fixed-Scope: used when scope is more important



May require rounding up to be an integer



1. Understand the role of a project schedule
2. Understand how to develop a project schedule
3. Understand how to use a project schedule to monitor and track project progress
4. Understand agile planning principles



1. F. P. Brooks. The mythical man-month. In Essays on software engineering. Addison-Wesley, 1995.
2. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
3. Kenneth S. Rubin. Essential Scrum – A Practical Guide to the Most Popular Agile Process. Addison-Wesley, 2013.