



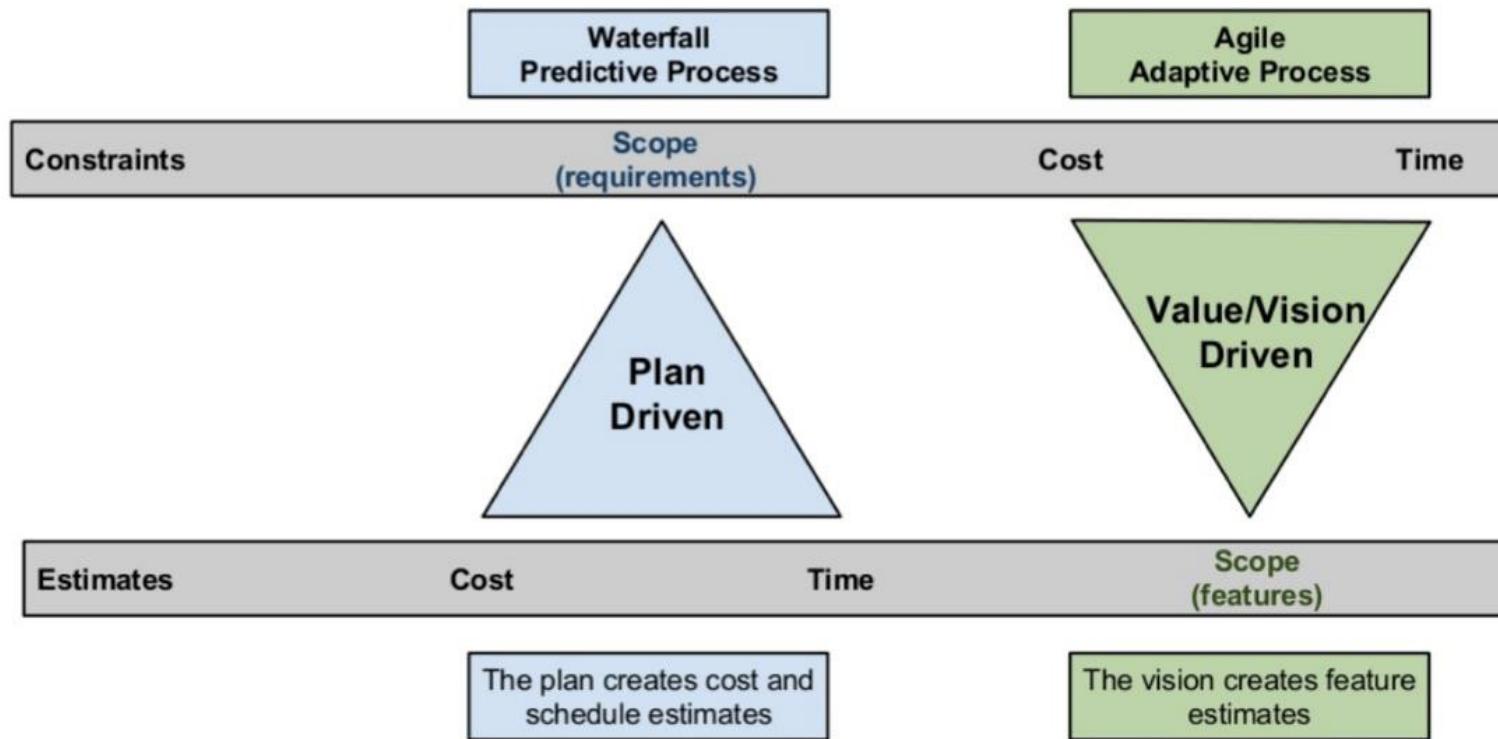
SWEN90016

Software Processes & Project Management

Project Planning and Scheduling



How to plan and control the schedule of software projects.

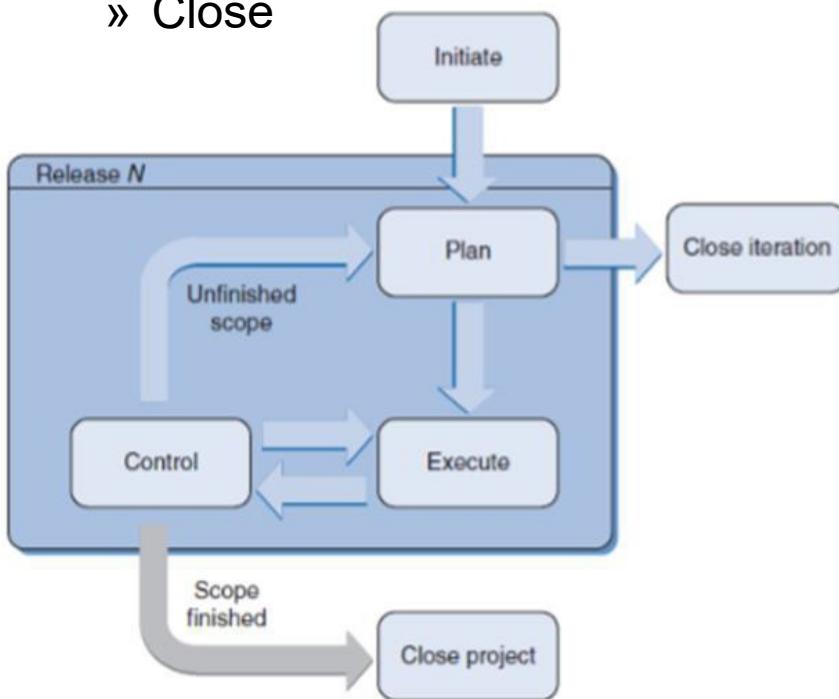




INTRODUCTION

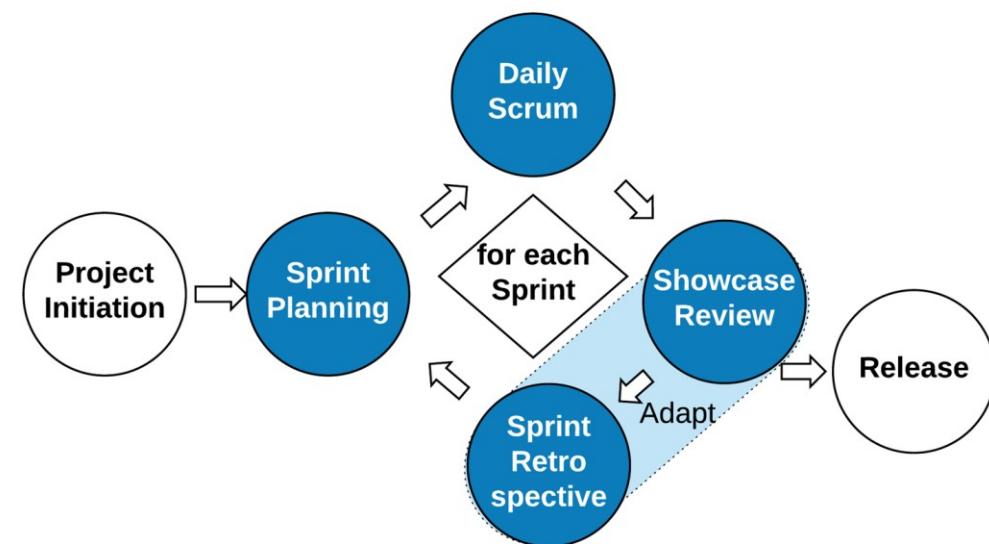
Formal PM Stages:

- » Initiate
- » Plan
- » Execute
- » Monitor & Control
- » Close



Agile PM Stages:

- » Initiate
- » Sprint Plan
- » Scrum (or Sprint)
- » Review & Retrospective (or Adapt)
- » Release





What steps are involved in developing a project schedule?

Identify tasks

(breakdown to tasks)

Identify dependencies

(task network tool)

Estimate duration & resources

(staff, hardware, artefacts)

Construct **schedule**

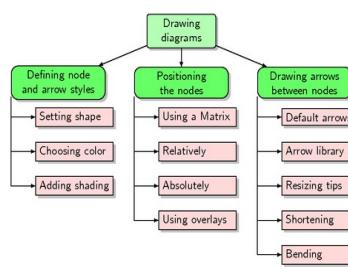
(allocate dates, resources)

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**

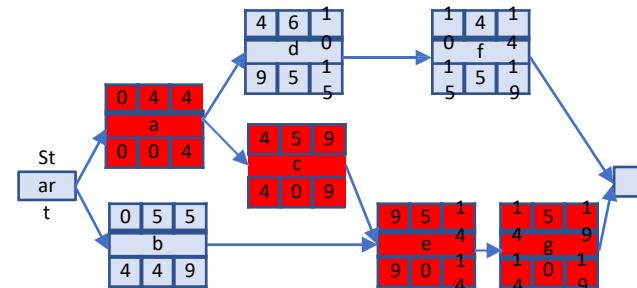


1. Work Breakdown Structure

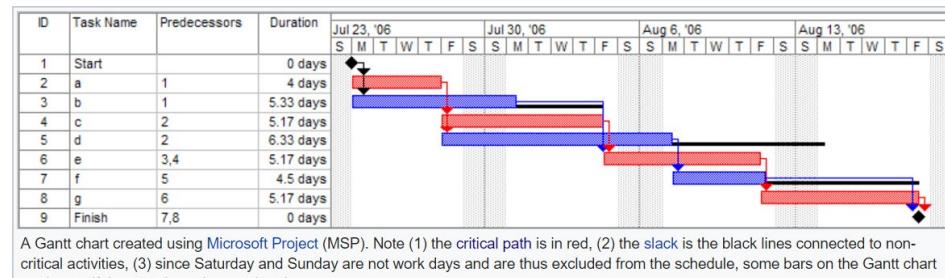
Redecorate Room
Prepare materials
<ul style="list-style-type: none">Buy paintBuy a ladderBuy brushes/rollersBuy wallpaper remover
Prepare room
<ul style="list-style-type: none">Remove old wallpaperRemove detachable decorationsCover floor with old newspapersCover electrical outlets/switches with tapeCover furniture with sheets
Paint the room
Clean up the room
<ul style="list-style-type: none">Dispose or store leftover paintClean brushes/rollersDispose of old newspapersRemove covers



2. PERT Chart



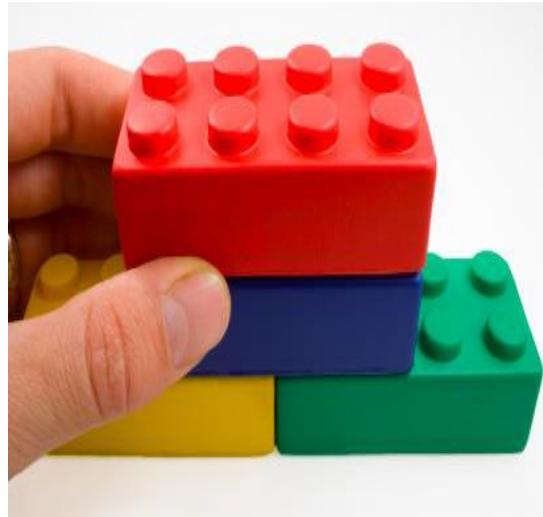
3. Gantt Chart



how to plan
the schedule



WORK BREAKDOWN



Activity	Work Breakdown
1.	Concept Phase Concept Planning Initial Research Problem definition with client Initial Project Plan
2.	Requirements Requirements Iteration 1 2.1.1 Requirement Elicitation 2.1.2 Requirements Analysis 2.1.3 Requirement Model Requirements Iteration 2 2.2.1 Requirement Elicitation 2.2.2 Requirements Analysis 2.2.3 Requirement Model Requirements Specification Requirements Validation Requirements Sign-off
3.	Project Planning Technological Risk Assessment



Identify Dependencies

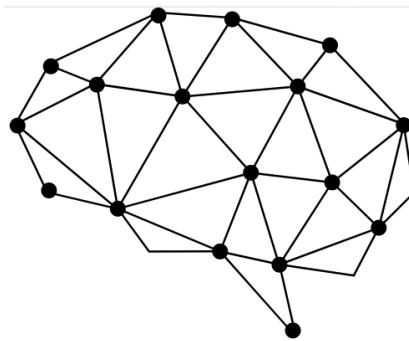
WORKSHEET 1

	Activity	Work Breakdown	Dependencies predecessor	Duration
1.	1.1 1.2 1.3 1.4	Concept Phase Concept Planning Initial Research Problem definition with client Initial Project Plan	1.1, 1.2, 1.3	1 4 2 1
2.	2.1 2.2 2.3 2.4 2.5	Requirements Requirements Iteration 1 2.1.1 Requirement Elicitation 2.1.2 Requirements Analysis 2.1.3 Requirement Model Requirements Iteration 2 2.2.1 Requirement Elicitation 2.2.2 Requirements Analysis 2.2.3 Requirement Model Requirements Specification Requirements Validation Requirements Sign-off	1.4 2.1.1 2.1.2 2.1.2 2.2.1 2.2.2 2.2.3 2.3 3.1, 2.4	2 3 3 3 3 4 5 4 4
3.	3.1	Project Planning Technological Risk Assessment	2.1.2	4



INDIVIDUAL WORK

Develop a task network
(activity on node)
given dependencies



	activity	predecessor	duration
1	1.1		1
2	1.2		4
3	1.3		2
4	1.4	1.1 1.2 1.3	1
5	2.1.1	1.4	2
6	2.1.2	2.1.1	3
7	2.1.3	2.1.2	3
8	2.2.1	2.1.2	3
9	2.2.2	2.2.1	3
10	2.2.3	2.2.2	4
11	2.3	2.2.3	5
12	2.4	2.3	4
13	2.5	2.4 3.1	4
14	3.1	2.1.2	4



PERT: Program Evaluation & Review Technique

ES	Duration	EF
Task Name		
LS	Slack	LF

The activity node

Earliest start time (ES)
Duration in people days
Earliest finish time (EF)

Latest start time (LS)
Slack time
Latest finish time (LF)



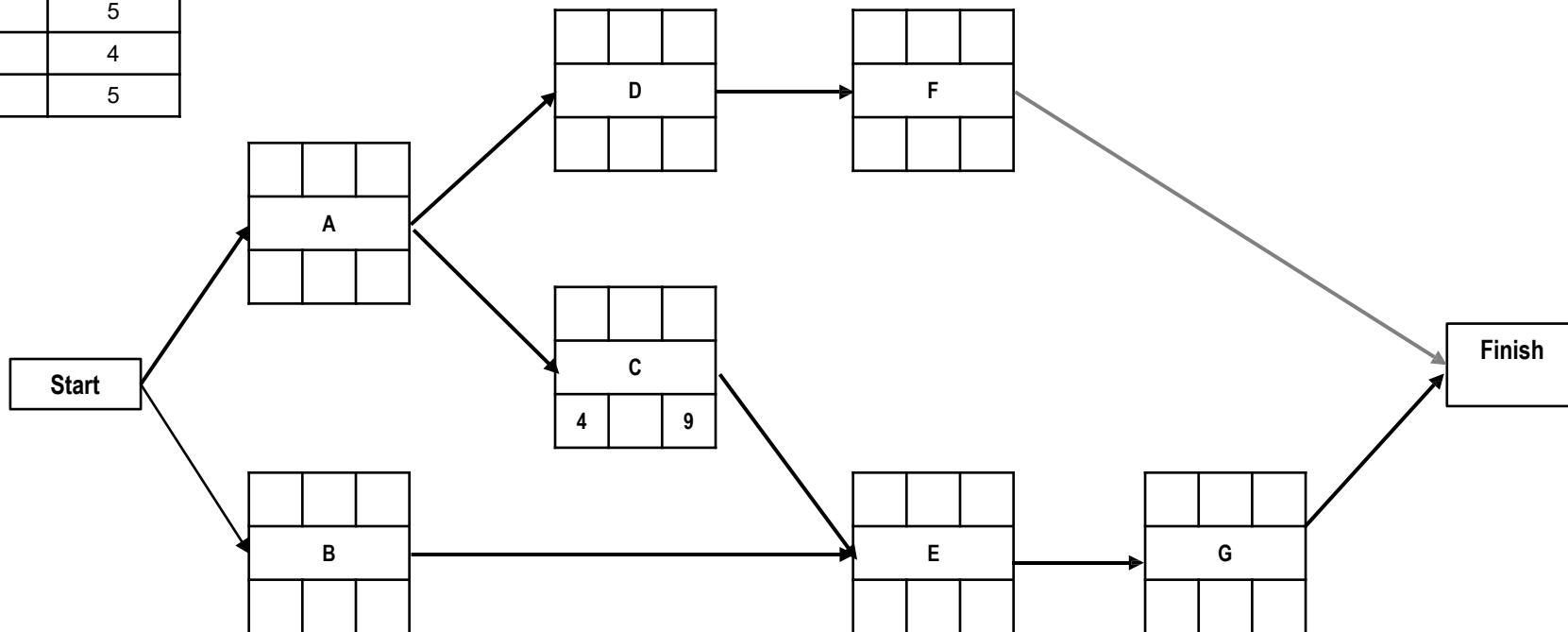
DATA SHEET

Show a PERT chart: use task durations & task network diagram

Activity	Duration
A	4
B	5
C	5
D	6
E	5
F	4
G	5

ES	Duration	EF
Task Name		
LS	Slack	LF

Task Network Diagram

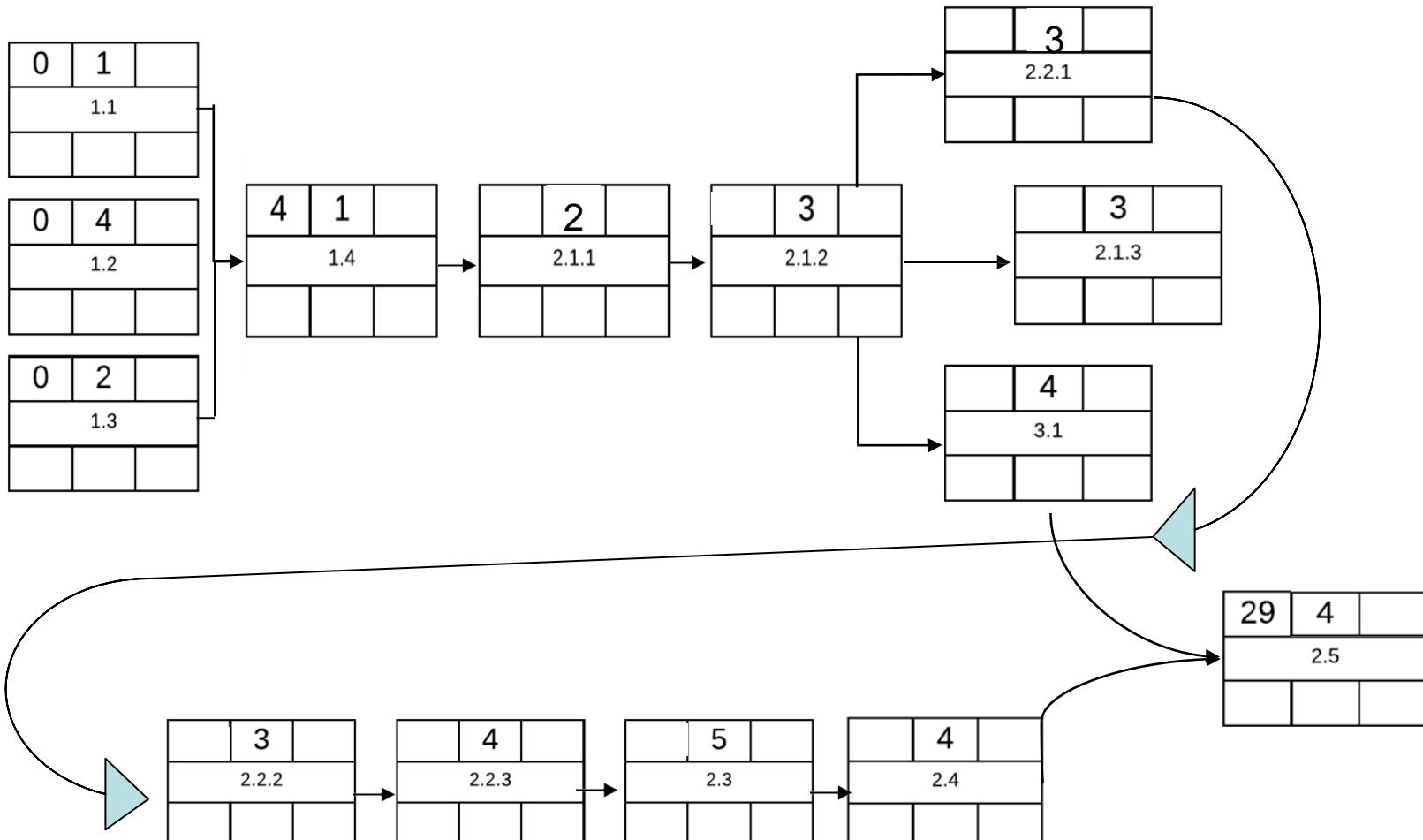




INDUSTRY LINK

Use duration estimates & task network to construct PERT chart

activ	durn
1.1	1
1.2	4
1.3	2
1.4	1
2.1.1	2
2.1.2	3
2.1.3	3
2.2.1	3
2.2.2	3
2.2.3	4
2.3	5
2.4	4
2.5	4
3.1	4

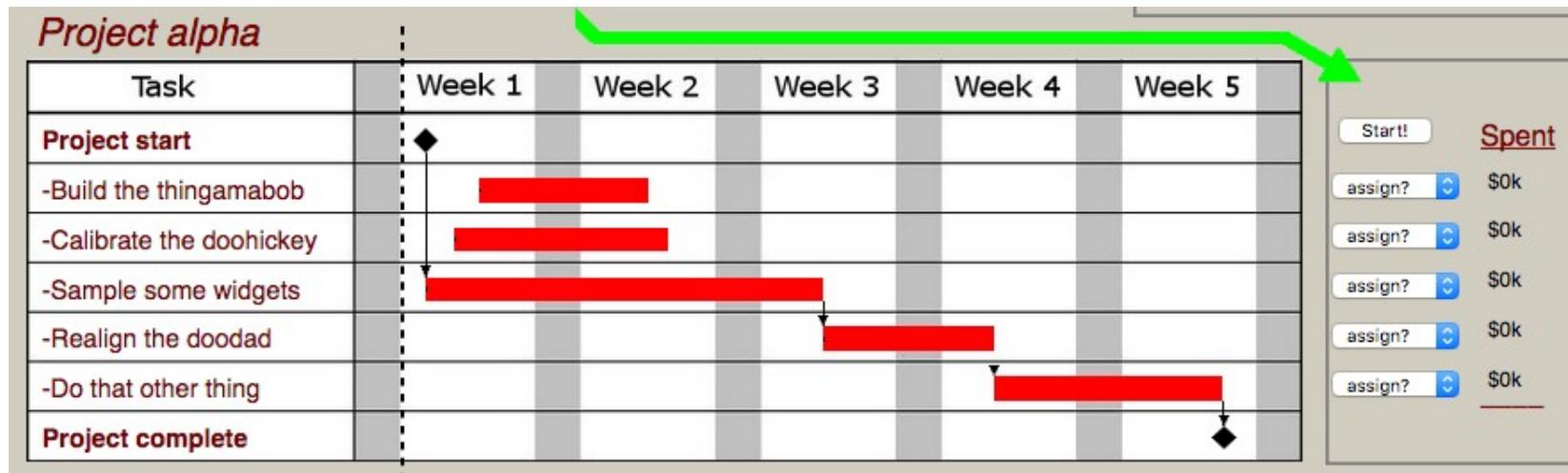




MANAGEMENT

Play the Project Management Game:

<http://thatpmgame.com/>

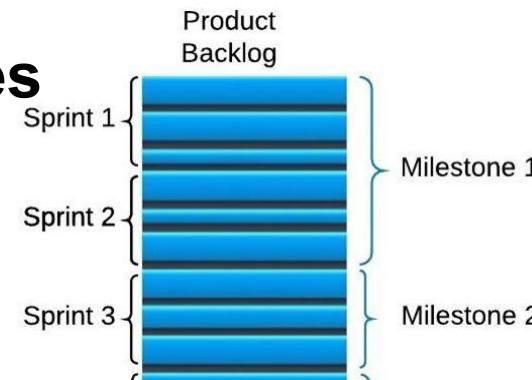
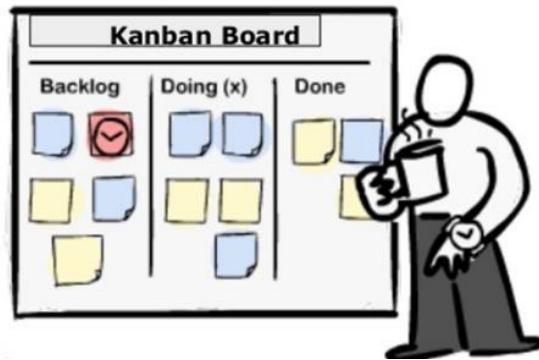


Use a Gantt chart to assign staff to various tasks.
Is the project completed on time and on budget?



Project Scheduling

1. Product Backlog with milestones



2. Sprint Backlog on Kanban board

3. Burndown Charts

how to plan
the schedule

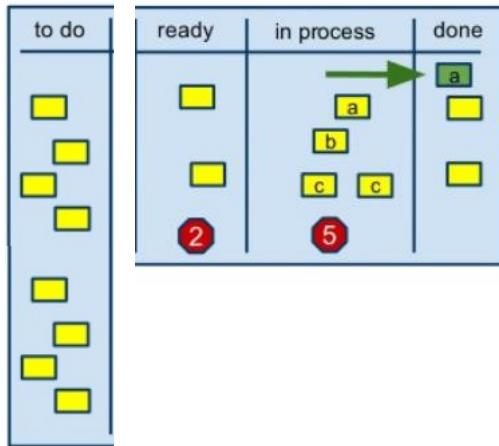


Velocity and Visual Board

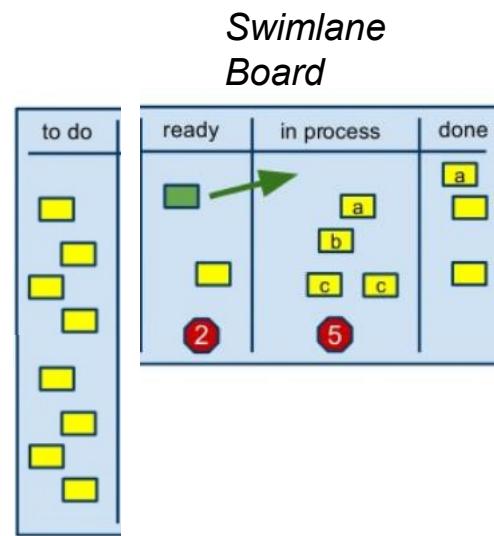
QUESTION

How many User Stories are “**done**” over the time-boxed Sprint?

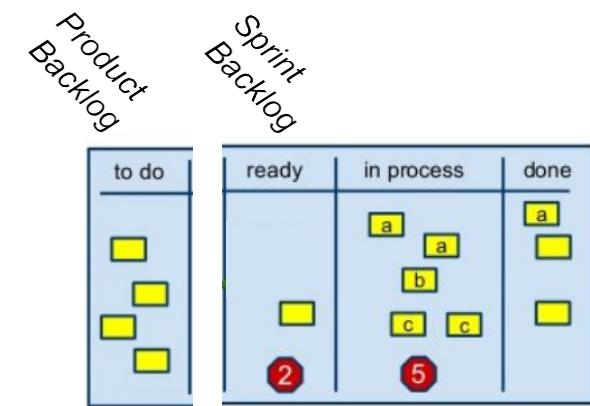
- Only count 100% complete stories
- Predict when the release milestones will be reached



Team member A completes code for a card and moves it to “done”



Team member A “pulls” a new card from “ready” and moves it to “doing”

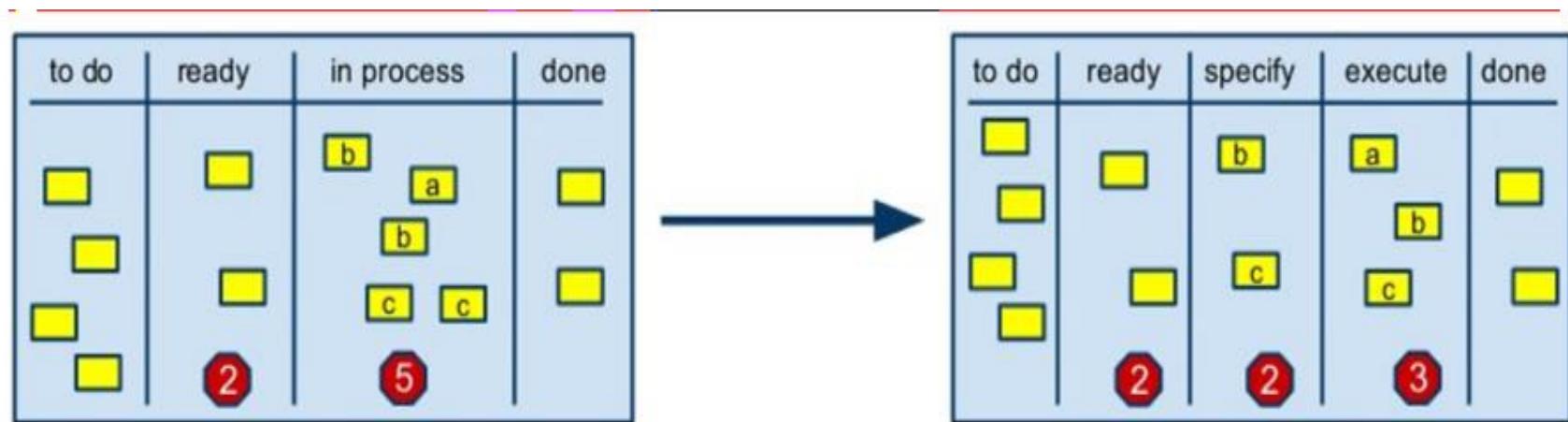


The Product Owner selects the next priority set of cards (Sprint Backlog) and moves it to “ready”



Velocity determines when dev team can deliver

- Dev team velocity emerges over a number of Sprints
- Predict when the release milestones will be reached





Velocity determines the slope of the BurnDown charts

- The Scrum master can track remaining effort
- Predict when the release milestones will be reached





Thank You!





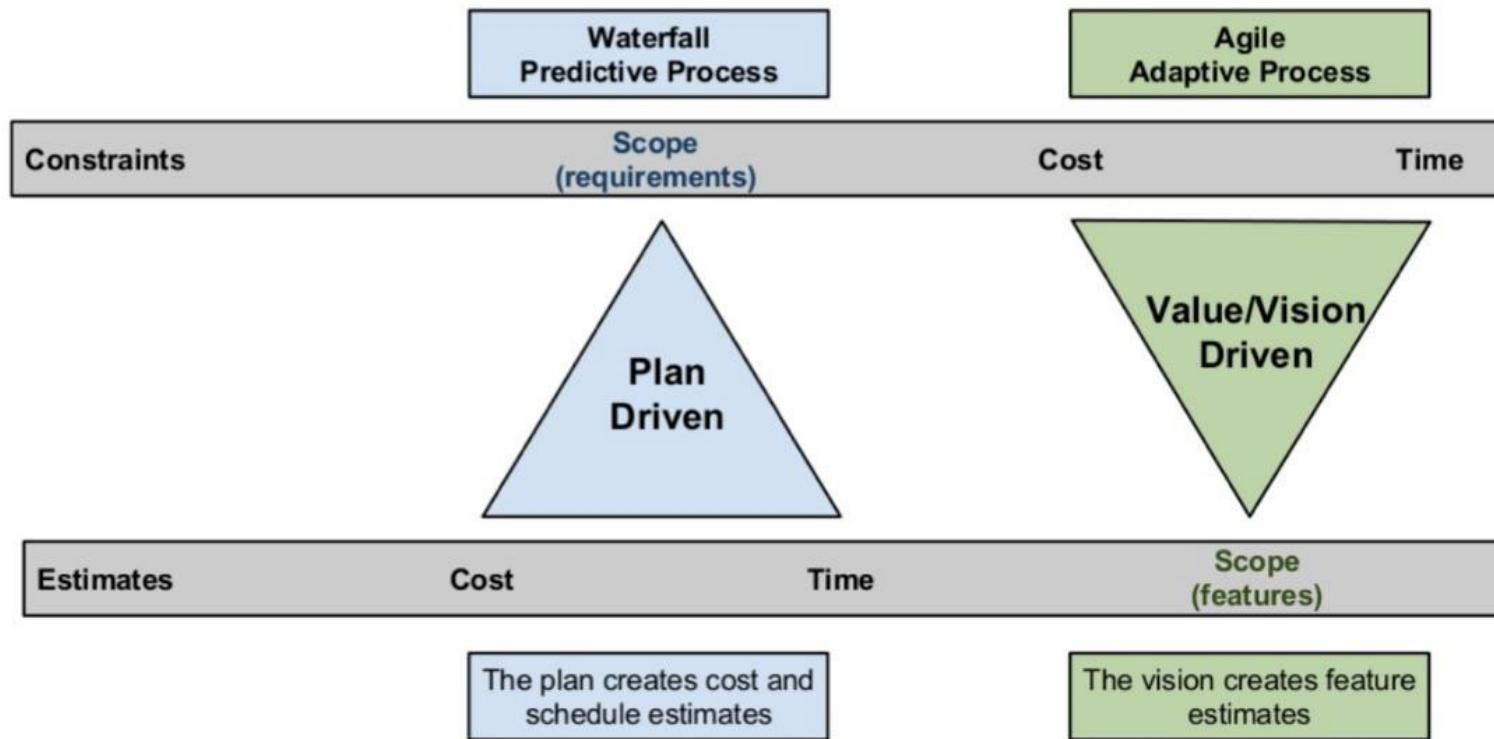
SWEN90016

Software Processes & Project Management

Project Planning and Scheduling



How to plan and control the schedule of software projects.

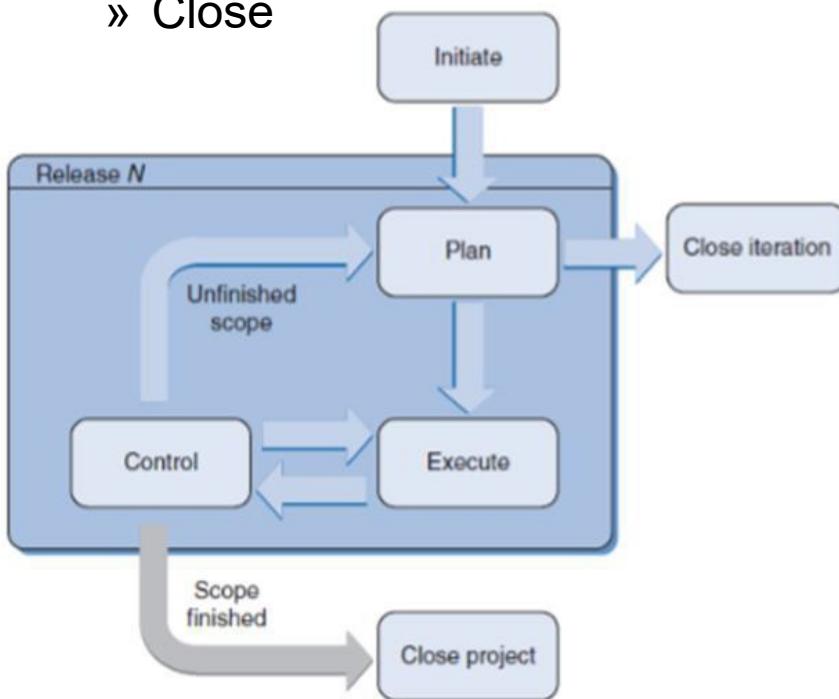




INTRODUCTION

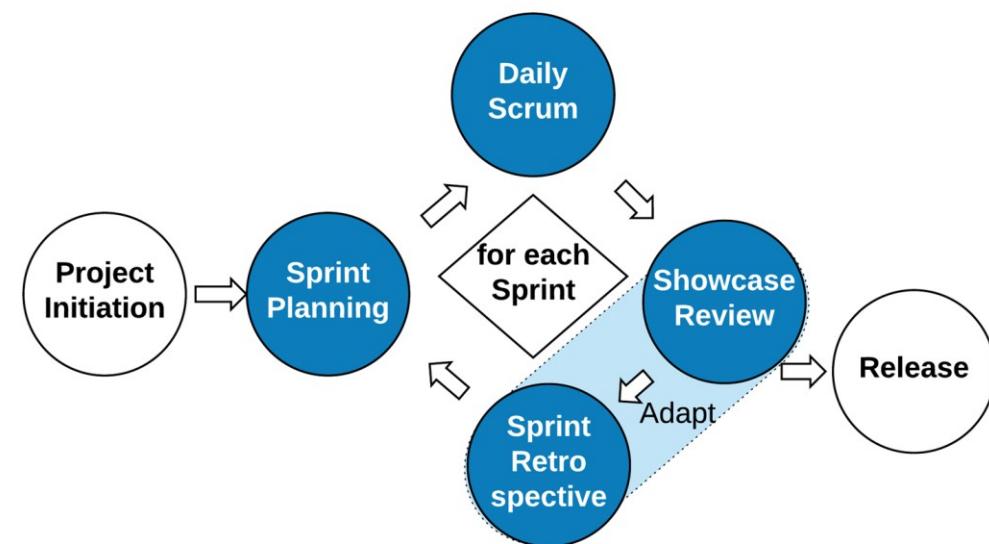
Formal PM Stages:

- » Initiate
- » Plan
- » Execute
- » Monitor & Control
- » Close



Agile PM Stages:

- » Initiate
- » Sprint Plan
- » Scrum (or Sprint)
- » Review & Retrospective (or Adapt)
- » Release





What steps are involved in developing a project schedule?

Identify tasks

(breakdown to tasks)

Identify dependencies

(task network tool)

Estimate duration & resources

(staff, hardware, artefacts)

Construct **schedule**

(allocate dates, resources)

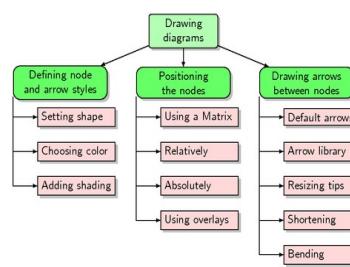
1. Breakdown the task into small chunks you can deal with – **Work Breakdown Structure (WBS)**
2. Identify the **interdependencies** between the broken down tasks and develop a **task network**
3. Estimate the **effort** and the **time allocation** for each task
4. **Allocate resources** for tasks and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



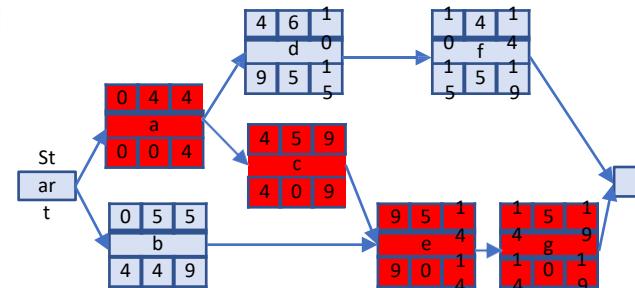
INSTITUTE FOR

1. Work Breakdown Structure

Redecorate Room
Prepare materials
<ul style="list-style-type: none">Buy paintBuy a ladderBuy brushes/rollersBuy wallpaper remover
Prepare room
<ul style="list-style-type: none">Remove old wallpaperRemove detachable decorationsCover floor with old newspapersCover electrical outlets/switches with tapeCover furniture with sheets
Paint the room
Clean up the room
<ul style="list-style-type: none">Dispose or store leftover paintClean brushes/rollersDispose of old newspapersRemove covers

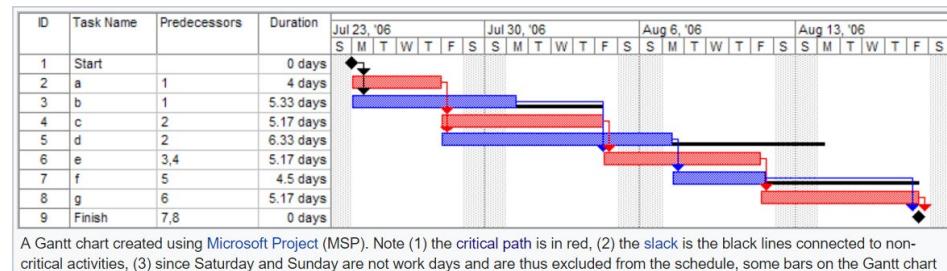


2. PERT Chart



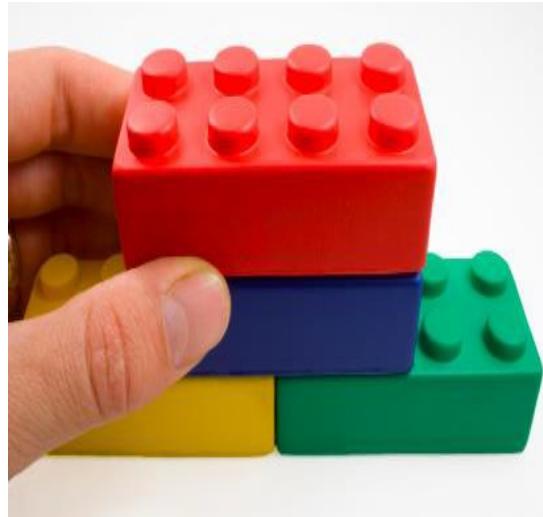
how to plan
the schedule

3. Gantt Chart





WORK BREAKDOWN



Activity	Work Breakdown
1.	Concept Phase Concept Planning Initial Research Problem definition with client Initial Project Plan
2.	Requirements Requirements Iteration 1 2.1.1 Requirement Elicitation 2.1.2 Requirements Analysis 2.1.3 Requirement Model Requirements Iteration 2 2.2.1 Requirement Elicitation 2.2.2 Requirements Analysis 2.2.3 Requirement Model Requirements Specification Requirements Validation Requirements Sign-off
3.	Project Planning Technological Risk Assessment



Identify Dependencies

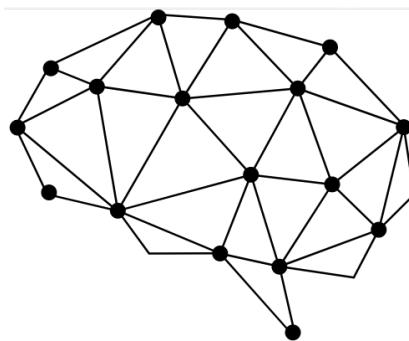
WORKSHEET 1

	Activity	Work Breakdown	Dependencies predecessor	Duration
1.	1.1 1.2 1.3 1.4	Concept Phase Concept Planning Initial Research Problem definition with client Initial Project Plan	1.1, 1.2, 1.3	1 4 2 1
2.	2.1 2.2 2.3 2.4 2.5	Requirements Requirements Iteration 1 2.1.1 Requirement Elicitation 2.1.2 Requirements Analysis 2.1.3 Requirement Model Requirements Iteration 2 2.2.1 Requirement Elicitation 2.2.2 Requirements Analysis 2.2.3 Requirement Model Requirements Specification Requirements Validation Requirements Sign-off	1.4 2.1.1 2.1.2 2.1.2 2.2.1 2.2.2 2.2.3 2.3 3.1, 2.4	2 3 3 3 3 4 5 4 4
3.	3.1	Project Planning Technological Risk Assessment	2.1.2	4



INDIVIDUAL WORK

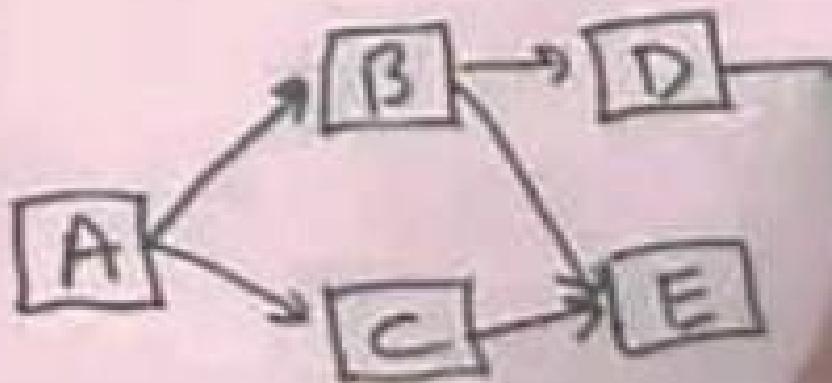
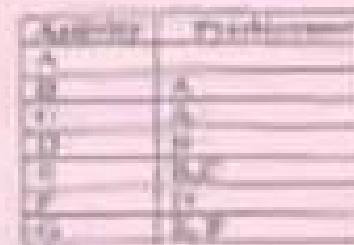
Develop a task network
(activity on node)
given dependencies



	activity	predecessor	duration
1	1.1		1
2	1.2		4
3	1.3		2
4	1.4	1.1 1.2 1.3	1
5	2.1.1	1.4	2
6	2.1.2	2.1.1	3
7	2.1.3	2.1.2	3
8	2.2.1	2.1.2	3
9	2.2.2	2.2.1	3
10	2.2.3	2.2.2	4
11	2.3	2.2.3	5
12	2.4	2.3	4
13	2.5	2.4 3.1	4
14	3.1	2.1.2	4



Given this summary information, draw a project network.

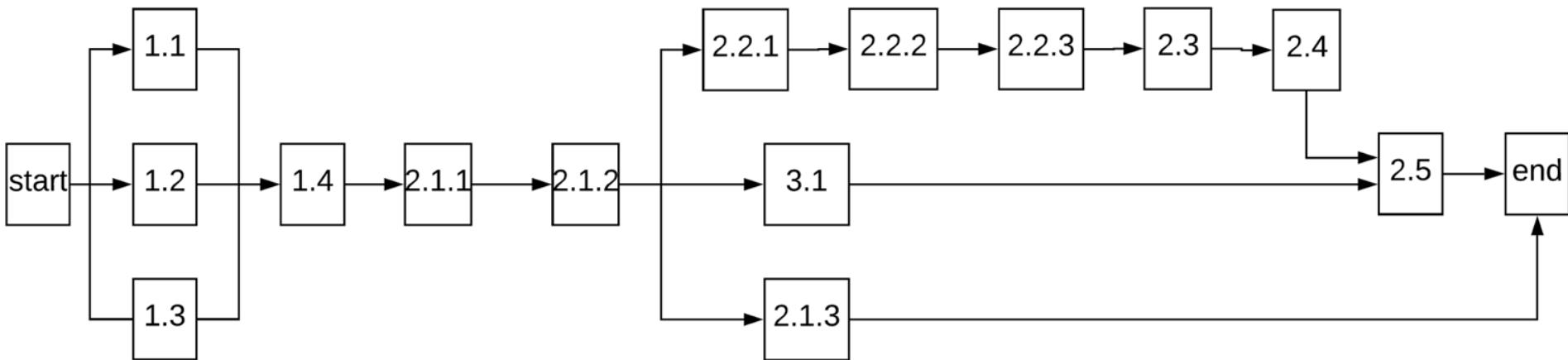




Task Network

Network Diagram

- Sequential nodes
- Few details





PERT: Program Evaluation & Review Technique

ES	Duration	EF
Task Name		
LS	Slack	LF

The activity node

Earliest start time (ES)
Duration in people days
Earliest finish time (EF)

Latest start time (LS)
Slack time
Latest finish time (LF)



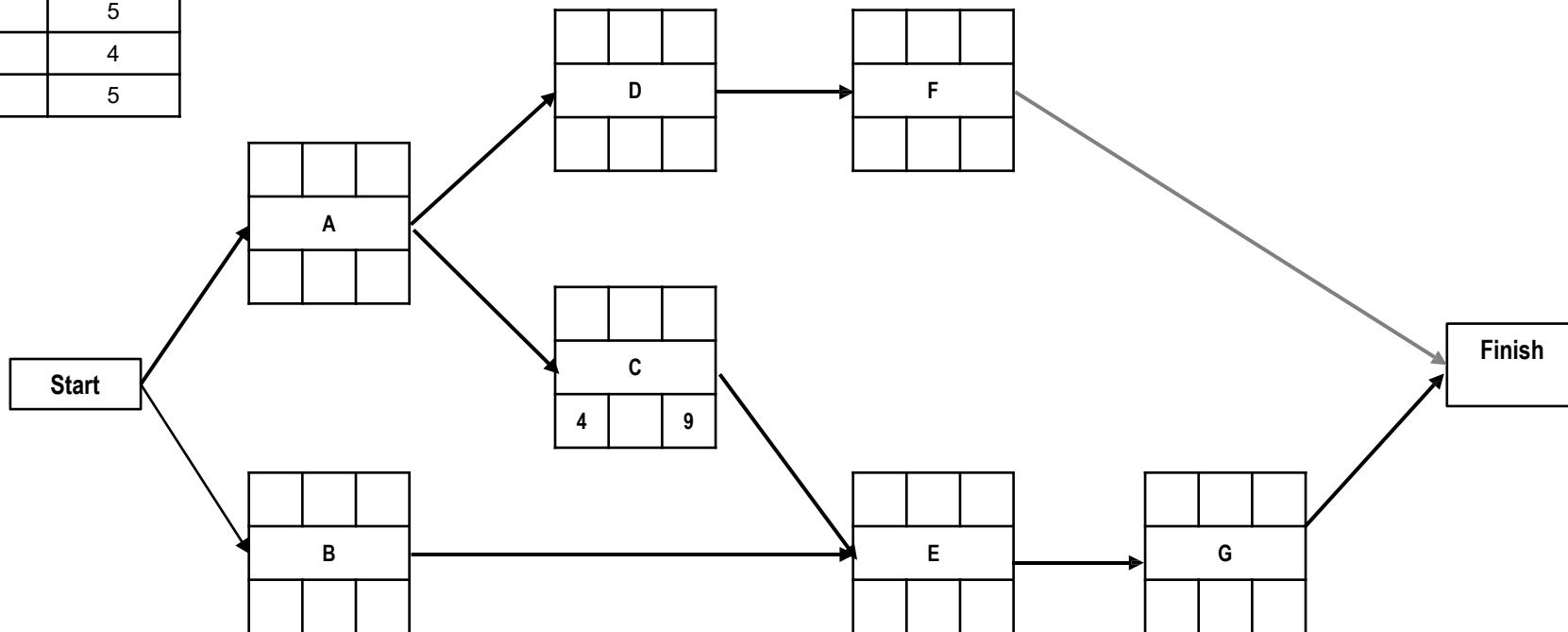
DATA SHEET

Show a PERT chart: use task durations & task network diagram

Activity	Duration
A	4
B	5
C	5
D	6
E	5
F	4
G	5

ES	Duration	EF
Task Name		
LS	Slack	LF

Task Network Diagram

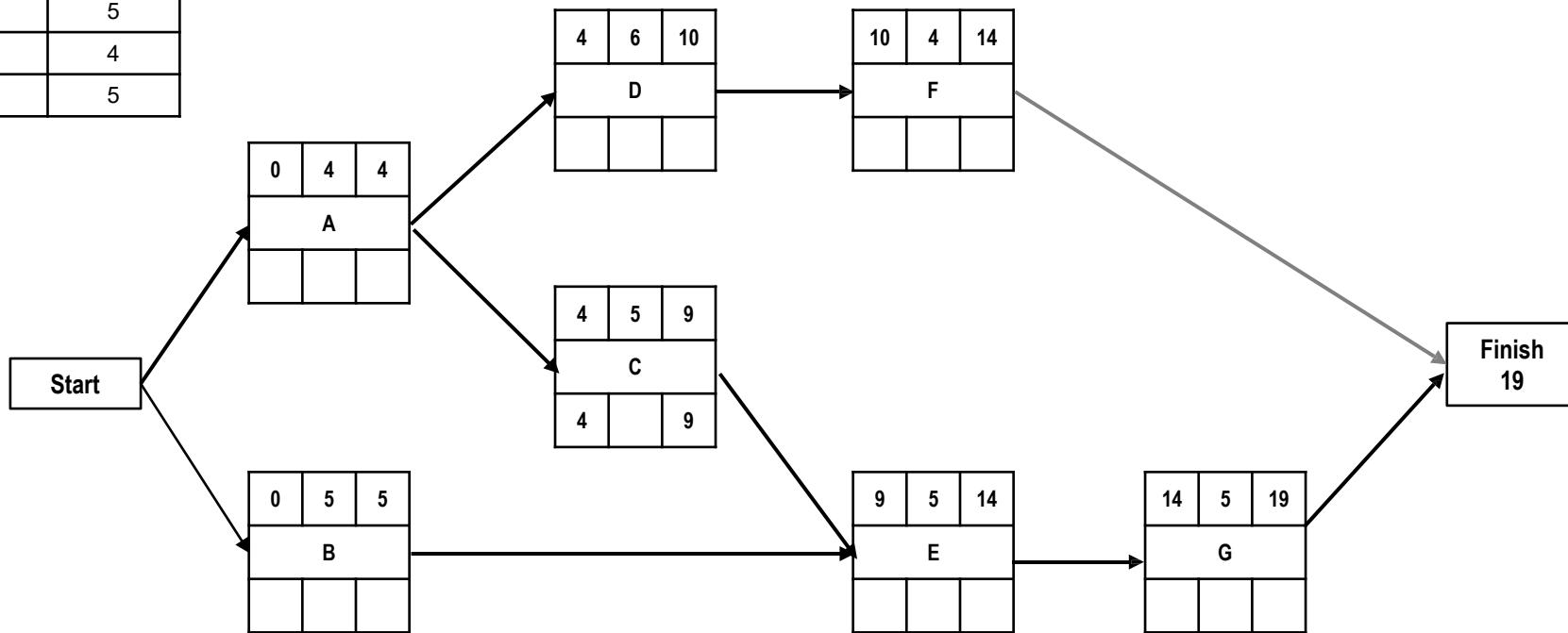




Forward Pass

Activity	Duration
A	4
B	5
C	5
D	6
E	5
F	4
G	5

ES	Duration	EF
Task Name		
LS	Slack	LF

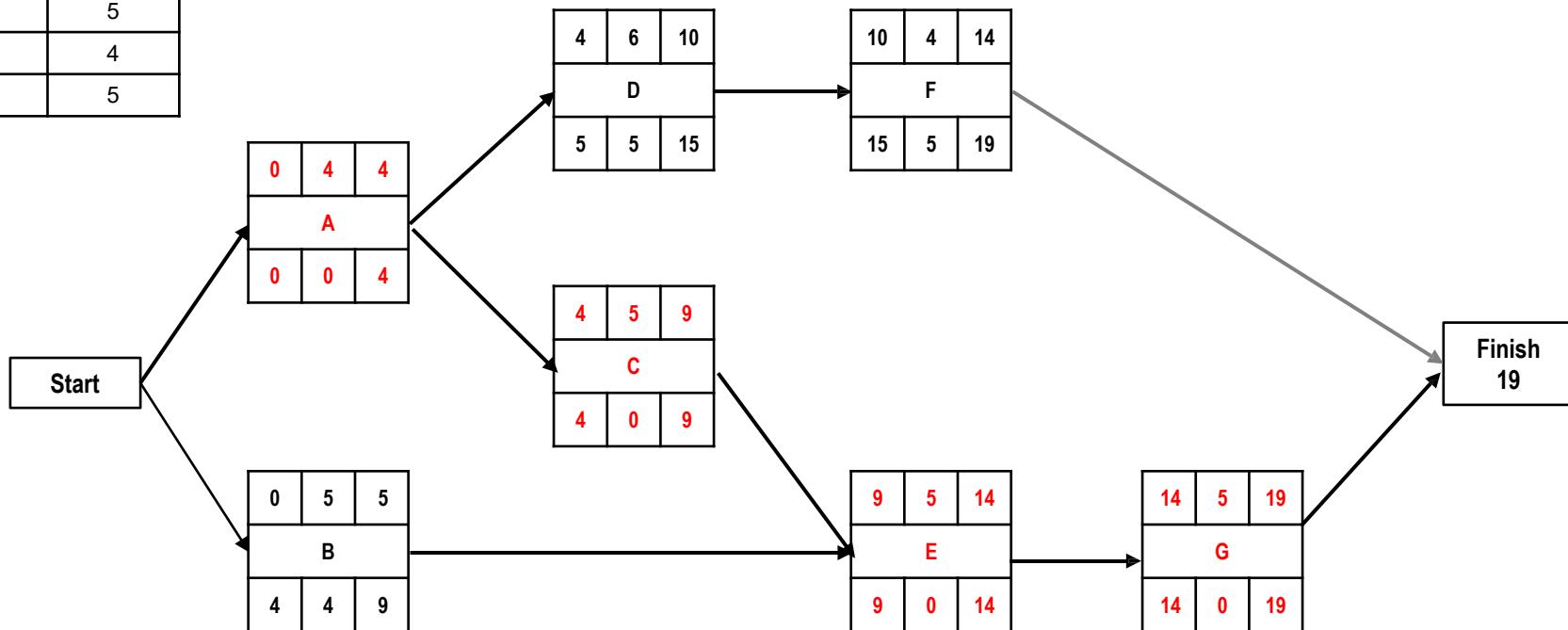




Slack

Activity	Duration
A	4
B	5
C	5
D	6
E	5
F	4
G	5

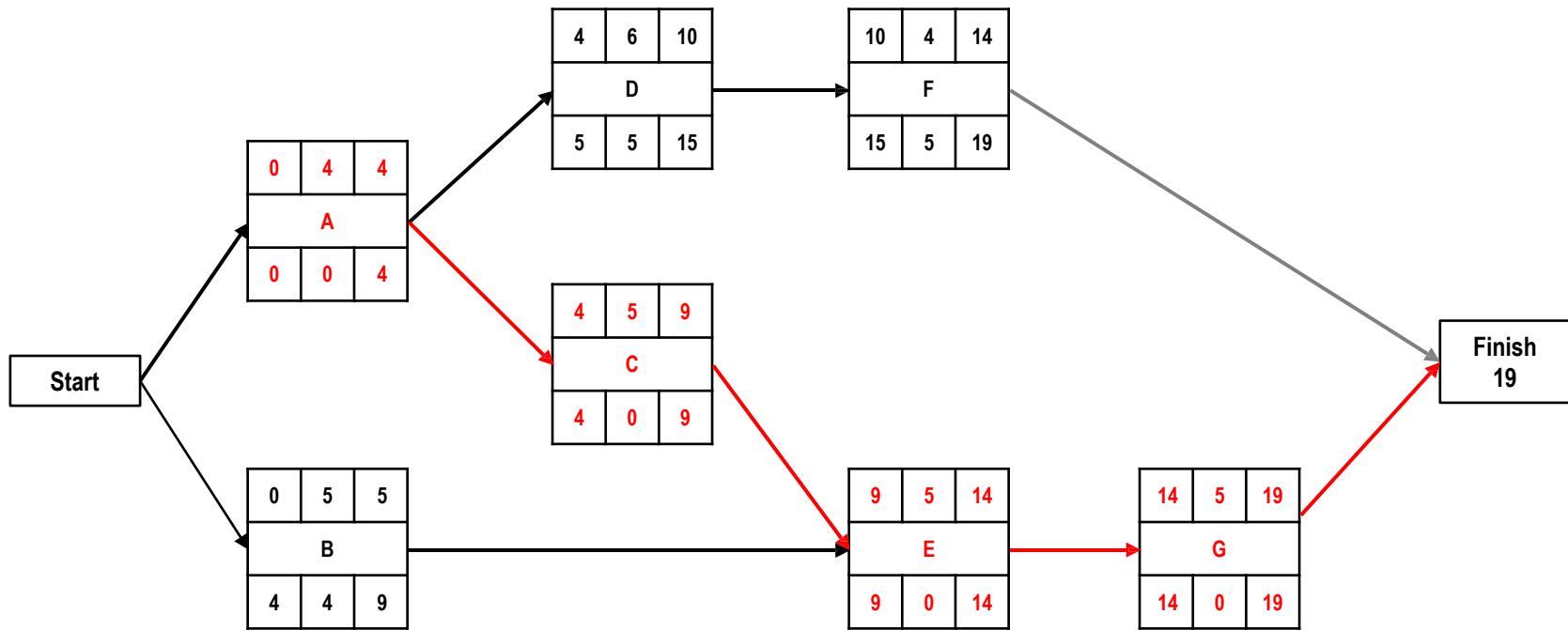
ES	Duration	EF
Task Name		
LS	Slack	LF





Critical Path

Critical Path = A + C + E + G

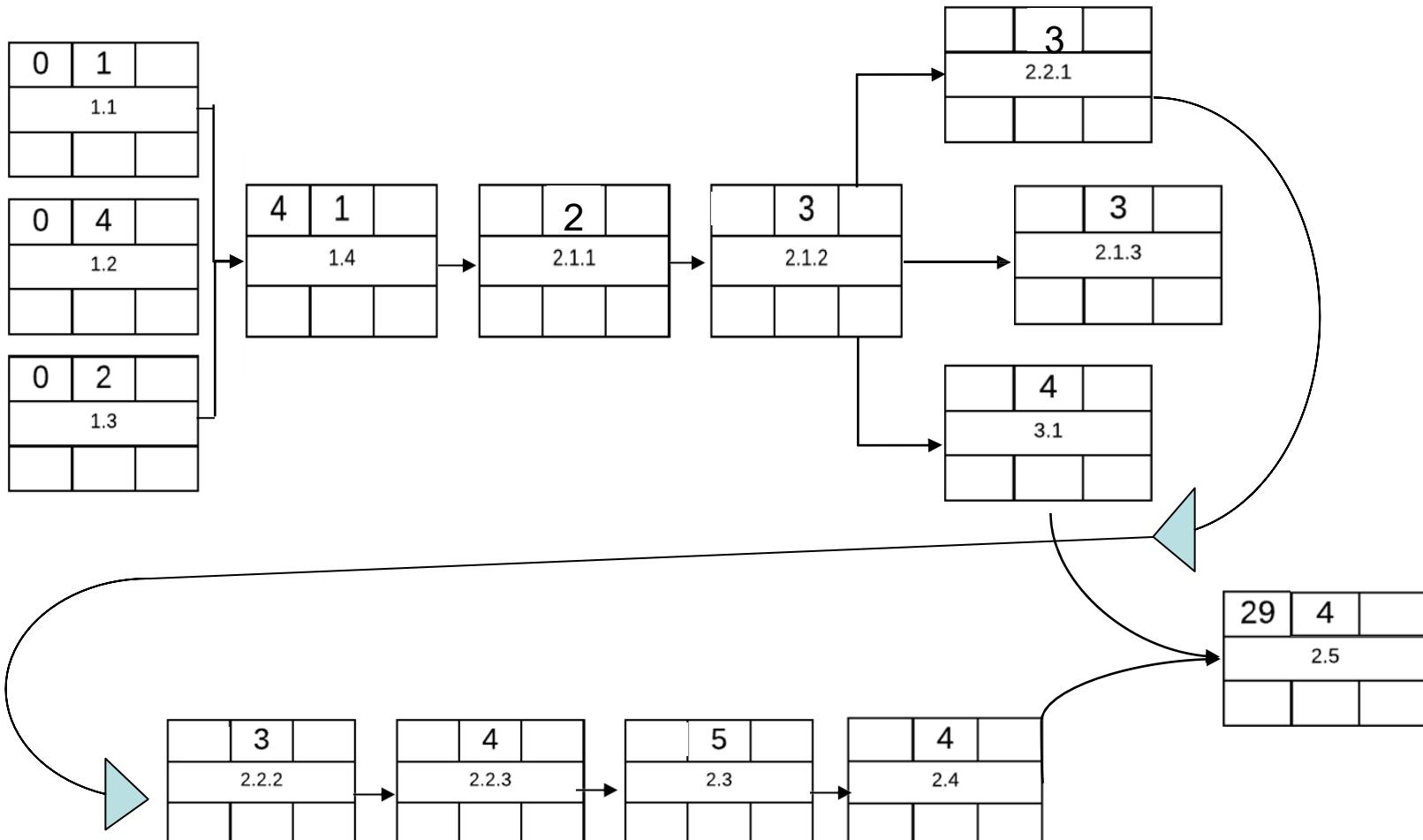


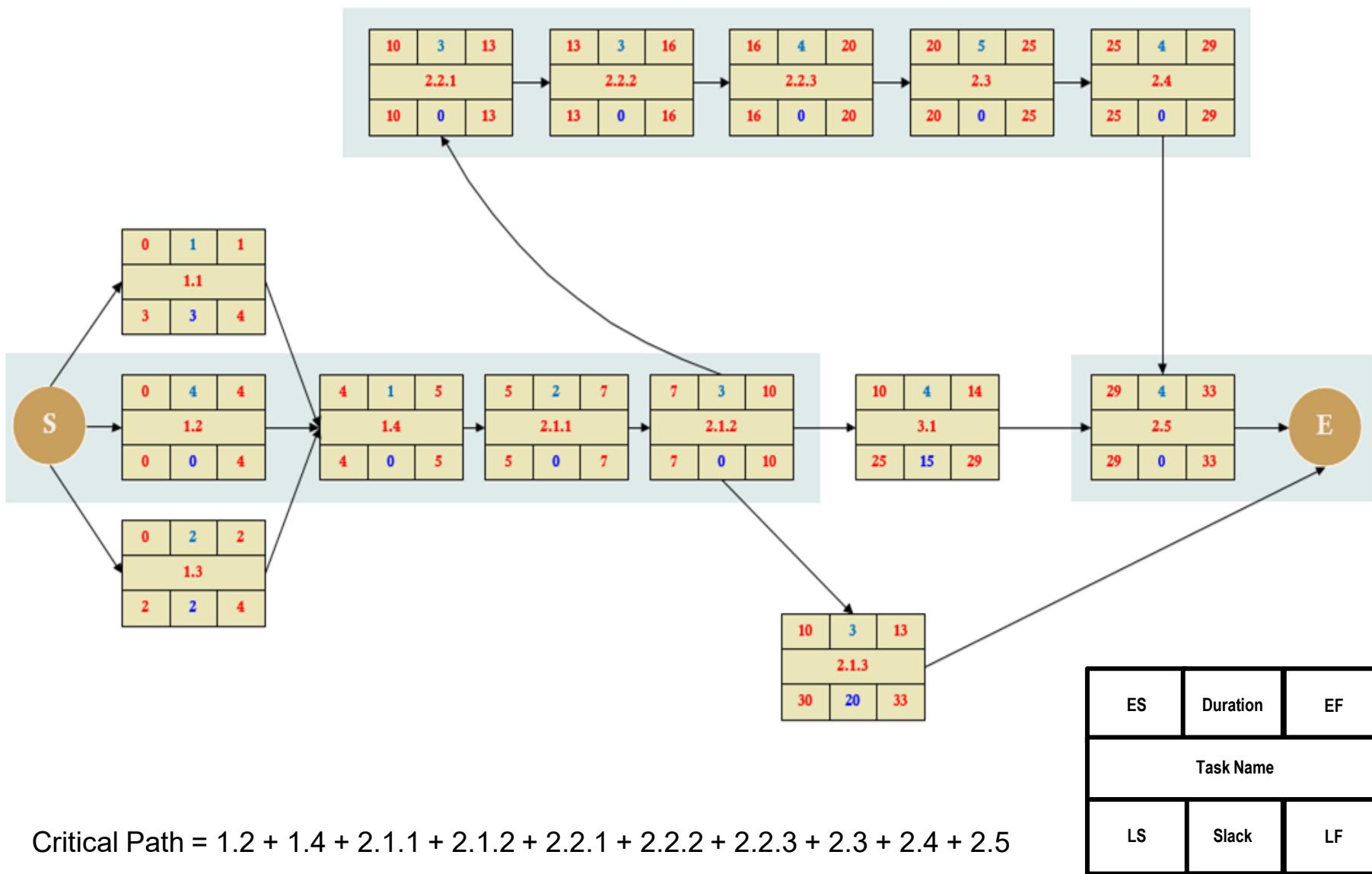


INDUSTRY LINK

Use duration estimates & task network to construct PERT chart

activ	durn
1.1	1
1.2	4
1.3	2
1.4	1
2.1.1	2
2.1.2	3
2.1.3	3
2.2.1	3
2.2.2	3
2.2.3	4
2.3	5
2.4	4
2.5	4
3.1	4



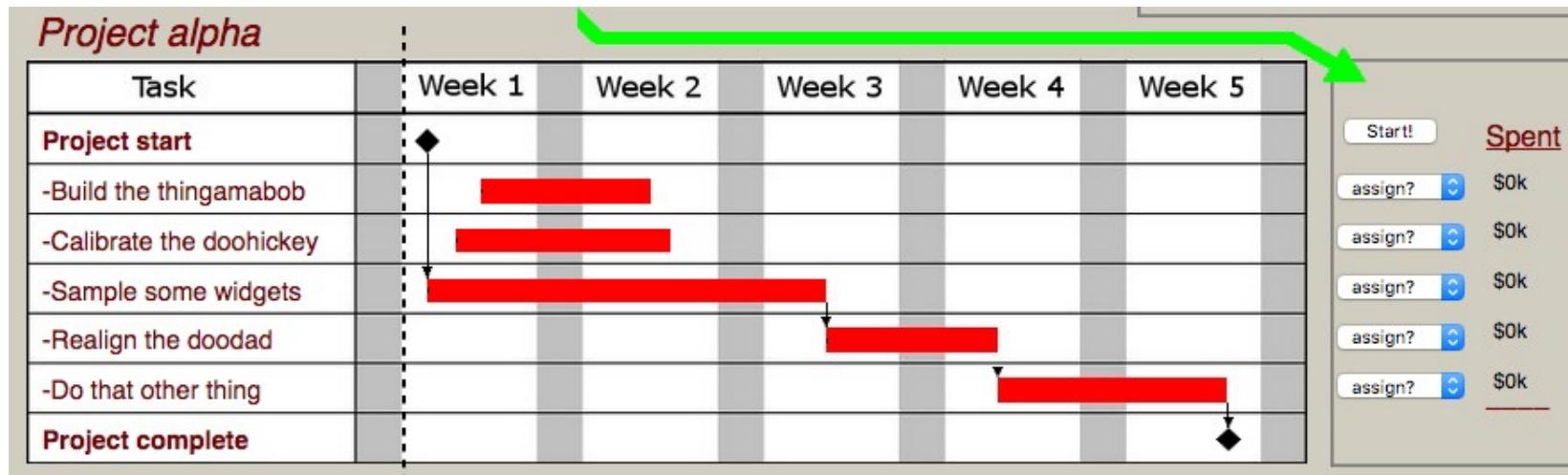




MANAGEMENT

Play the Project Management Game:

<http://thatpmgame.com/>

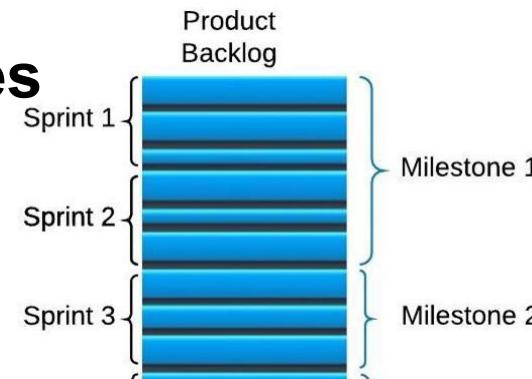
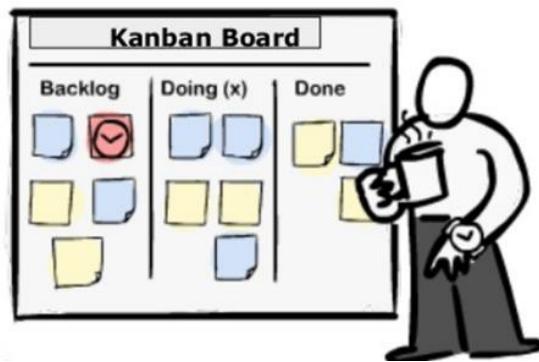


Use a Gantt chart to assign staff to various tasks.
Is the project completed on time and on budget?



Project Scheduling

1. Product Backlog with milestones



2. Sprint Backlog on Kanban board

3. Burndown Charts

how to plan
the schedule

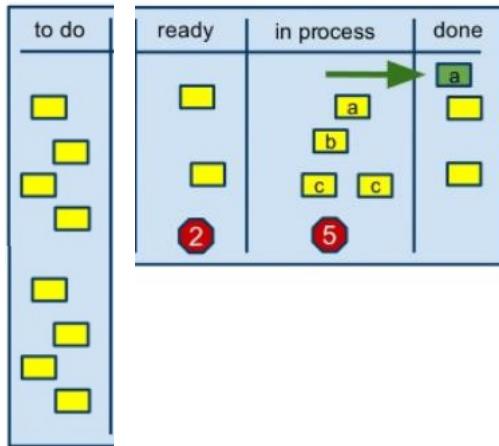


Velocity and Visual Board

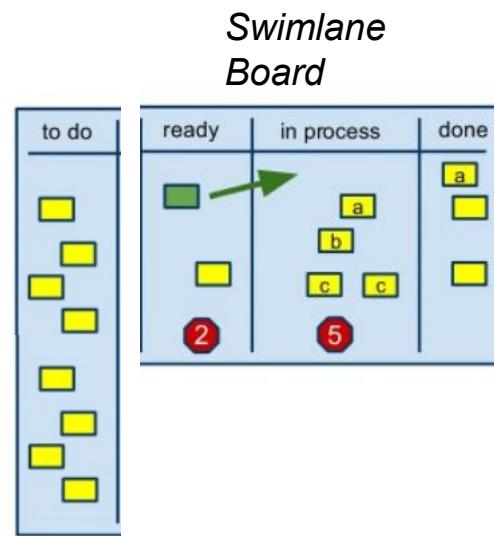
QUESTION

How many User Stories are “**done**” over the time-boxed Sprint?

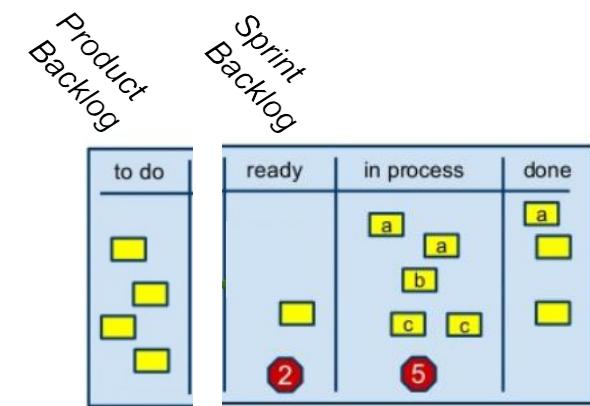
- Only count 100% complete stories
- Predict when the release milestones will be reached



Team member A completes code for a card and moves it to “done”



Team member A “pulls” a new card from “ready” and moves it to “doing”

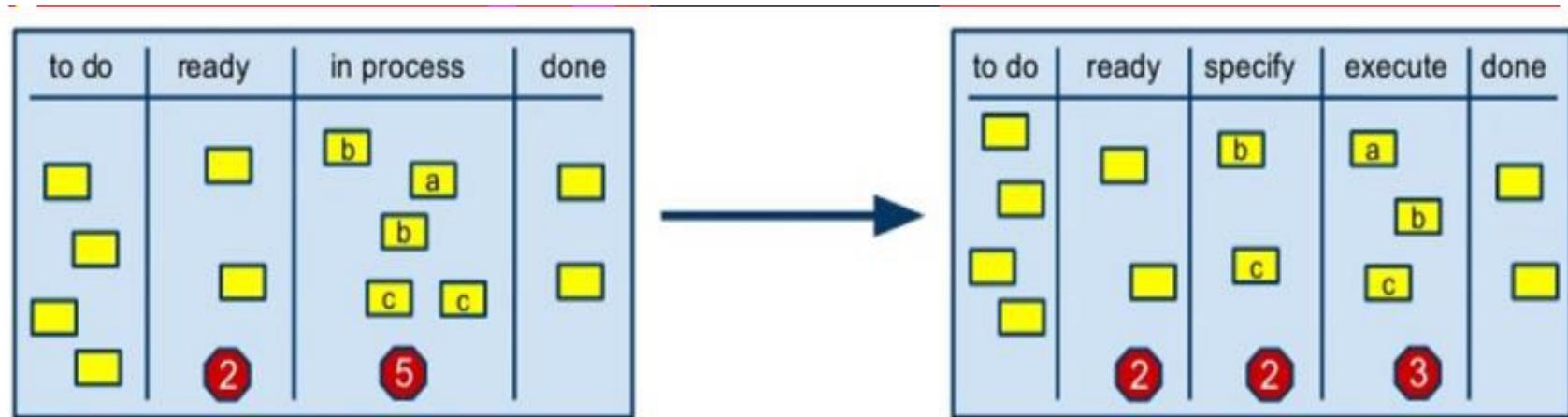


The Product Owner selects the next priority set of cards (Sprint Backlog) and moves it to “ready”



Velocity determines when dev team can deliver

- Dev team velocity emerges over a number of Sprints
- Predict when the release milestones will be reached

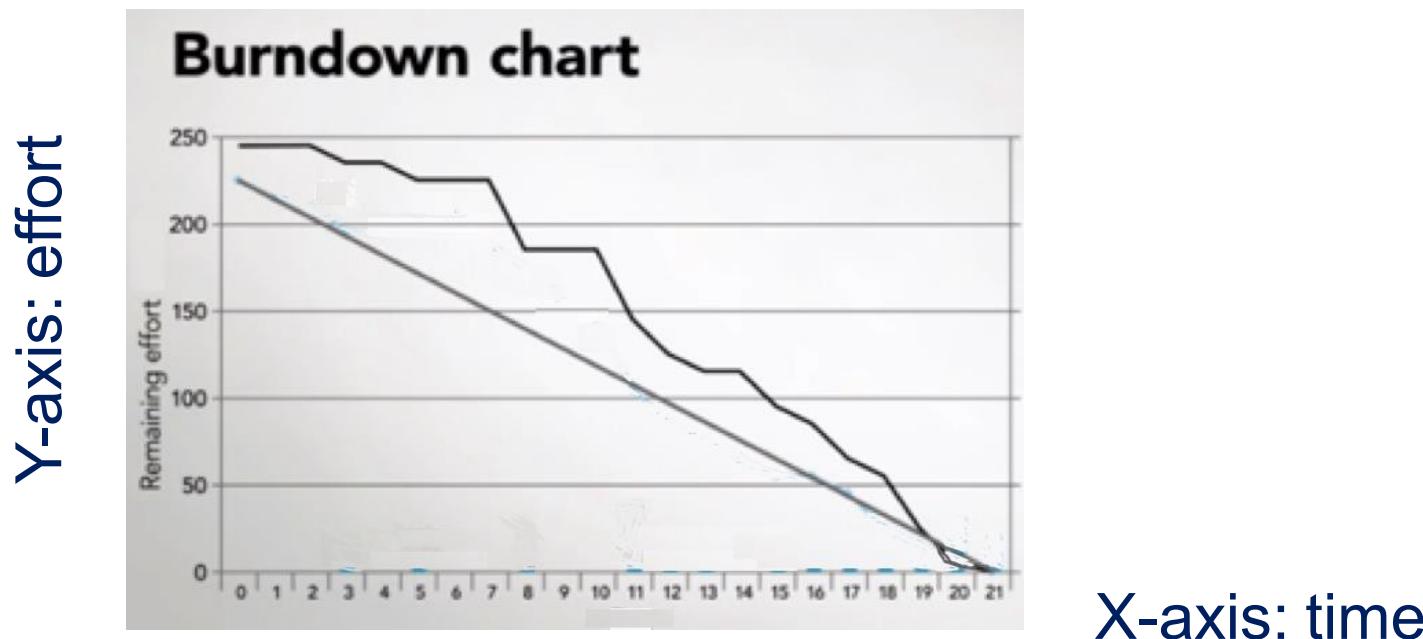




INNOVATION

Velocity determines the slope of the BurnDown charts

- The Scrum master can track remaining effort
- Predict when the release milestones will be reached





Thank You!





SWEN90016

Software Processes & Project Management

Risk Management

2021 – Semester 1
Tutorial 4



Understand Risk Management



for Language Research project



Recap: Challenges vs. Risks

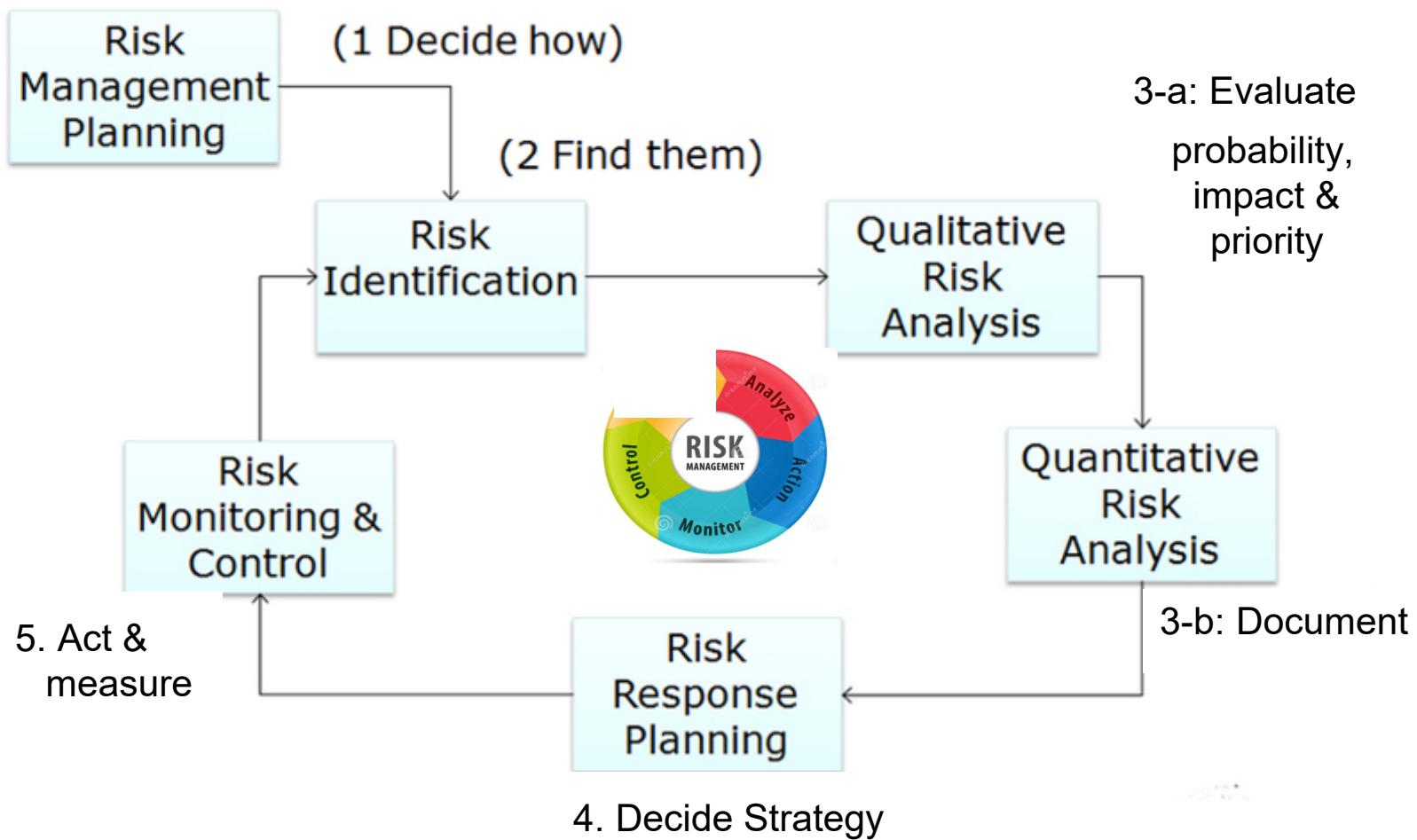
MIDDLEWARE

Challenge:



Risk:







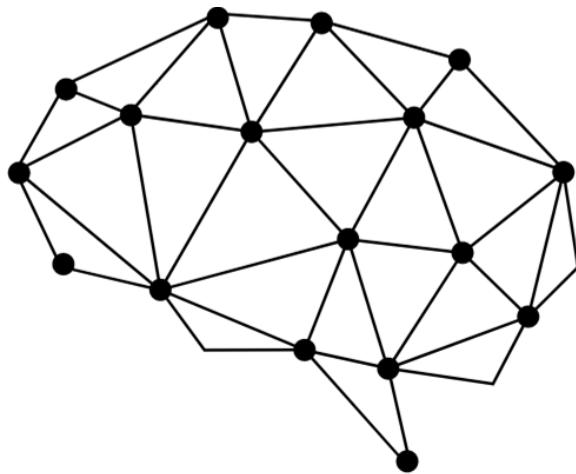
WILLING TO RISK

For the **Language Research** Case Study

Some key characteristics are:



Evaluate probability, impact & priority



Sift through all the risks in the project

Review and debate their importance

3. What are the **top three/four risks** to be controlled?

Document in a table.



Risk No	Risk description	Trigger event	Risk Owner	Consequence	Probability	Mitigation strategy
1	SMEs unavailable. SMEs are required to perform other duties while working on the project	Each team member has a number of hours assigned to the project each week	Ping Lu	Lack of team involvement will have an adverse affect on schedule	High (.9)	Immediate resolution: Hire temporary personnel for low skilled labour jobs while SME works on the project Monitor the mitigation strategy
2	Scope of the project is changing – scope management plan is too lenient	Middle management requiring changes to system after scope defined	Nick Lees	Will have an adverse affect on the scope of the project causing cost and time delays	High (.9)	Immediate resolution: Reevaluate the scope management plan and set guidelines in place Monitor the mitigation strategy



Strategic Risk Register – May 2006

Ref No.	Corporate Objective	Ref. No.	Risk Description	Risk Owner	Mitigation Control	Priority	Sources of Assurance
C2	To target resources & initiatives to overcome poverty and disadvantage	R1	Failure to achieve equality targets & improve community cohesion.	Corporate Equalities Group Environmental Services Director	1. Regular monitoring of Corporate Equalities Plan 2. Level 2 fully embedded by December 2006	M	Internal & external audit review. Consultation with minority groups.
		R2	Failure to deliver improvements in the benefits service.	CMT EMT OSCs Cabinet	1. Regular monitoring by TEN system 2. Quarterly reports to OSCs & Cabinet	H	Monitoring by DWP & BFI. Internal & external audit.
		R3	Costs of new concessionary fares scheme exceeding budget.	Assistant Director-Community Finance	1. Monitoring of costs, as part of integrated performance management report. 2. Quarterly reports to OSCs & Cabinet.	M	Cabinet & OSC monitoring. Monitoring with other Warwickshire Districts.
		R4	Failure to deliver major improvements in Camp Hill – reputation risk; loss of housing.	Cabinet Chief Executive	1. Monitoring by Project Board 2. External project management.	H	Pride in Camp Hill monitoring. Liaison with AWM & GOWM.
C3	To encourage the provision of new & improved housing to meet the needs of residents	R5	Failure to deliver continued improvements in Housing Services	Corporate Services Director Assistant Director-Housing	1. Monitoring of Improvement Plan.	M	GOWM monitoring. Housing inspectorate.
		R6	Failure to achieve the 'Decent Homes' standard for private sector housing.	Corporate Services Director Housing Portfolioholder	1. Stock Condition Survey.	M	Internal & external audit review. Performance indicators.
C6	To work in partnership to reduce information technology	R7	Failure to deliver continued	Chief Executive Assistant Director	1. Monitoring by Safer Communities Group	M	Annual external audit. Safer Communities



Risk Register: probability & time

Risk	Probability of Risk	Size of Loss (Days)	Risk Exposure (Days)
Backup and restore may require the inclusion of additional third-party products.	20%	15	3
The lack of scientifically relevant sample data impacts Partner A's ability to validate the product.	35%	20	7
There won't be time for Partner A to provide feedback on the format of Analysis reports, which means they could find the reports unacceptable during validation.	10%	5	0.5
Partner A employees are not available to validate the new features until too late in the process, limiting our ability to make additional releases that address any issues they might uncover.	20%	5	1
There won't be time in the QA process to validate, equally, on all browsers on all operating systems.	40%	5	2
Partner A may require more end-user documentation than has been provided.	25%	20	5
Ref: https://www.mountaingoatsoftware.com/blog/managing-risk-on-agile-projects-with-the-risk-burndown-chart	Exposure:	18.5	



4. Create a risk register to document the controlled risks for the language research study

Id	Risk Description
Risk 1	
Risk 2	
Risk 3	



5. Calculate: probability impact exposure

Risk	Probability of Risk	Size of Loss (Days)	Risk Exposure (Days)
Risk 1			
Risk 2		<i>the impact to the schedule if the risk did occur</i>	
Risk 3			

Risk **probability** = a measure between 0 and 1 inclusive

Risk **impact** = finite grade of 1-5 scale, such as:

(1) none; (2) minimal; (3) moderate; (4) severe; (5) catastrophic impact;
monetary cost or time cost?

Risk **exposure** = probability × impact



MIDDLEOURCE

Add Response strategy for handling threats & opportunities

THREAT RESPONSE	GENERIC STRATEGY	OPPORTUNITY RESPONSE
Avoid	Eliminate uncertainty	Exploit
Transfer	Allocate ownership	Share
Mitigate	Modify exposure	Enhance
Accept	Include in baseline	Ignore



<https://2020projectmanagement.com/news-and-events/risk-and-opportunity-management--a-3-dimensional-approach>



Monitor Process
Audit
Review
Status meetings

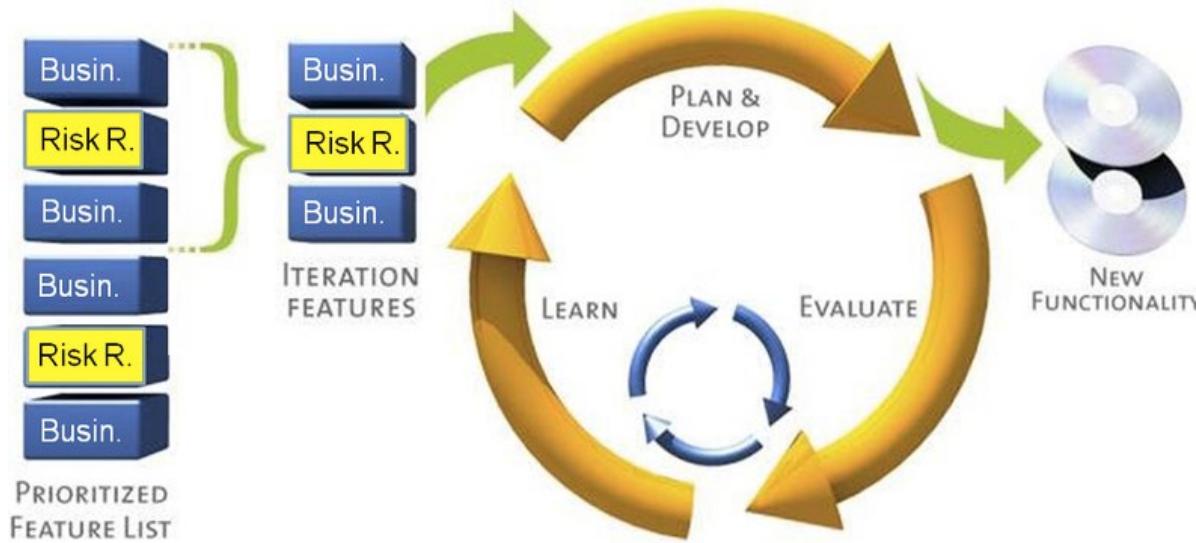


MELBOURNE

Identify – capture risks in Risk Register

Analyze – Product Backlog groomed, and priority given to all User Stories, including those which capture risk

leadinganswers.typepad.com/leading_answers/2007/09/agile-risk-mana.html



Respond – mitigate Risk in Sprint

Monitor – during Sprint Review, Retrospective & Planning



Sprint Review risk evaluation

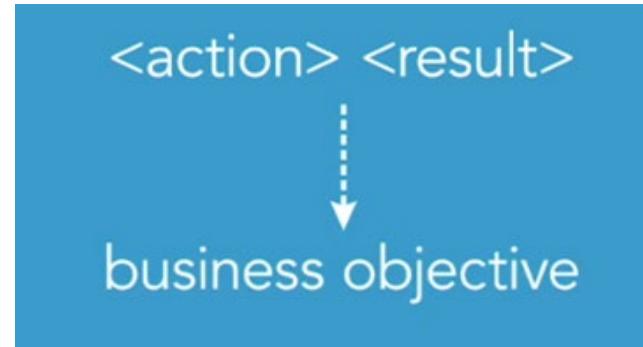
- Build small piece of working software with minimal features
- Showcase the product chunk to the stakeholders **early**
- Fail **fast** and as cheaply as possible, & get timely feedback
- Capture the **risk item** in the Product Backlog
- The Product Owner sets the priority of the **risk item**

What
Where
Who
When
Why



Sprint Review risk evaluation

- The format of a **risk item** in the Product Backlog can vary
- Optionally use Feature-Driven Development (**FDD**) syntax, (when the role is not obvious)



Example.

Risk 1: include request priority, for an effective booking service

www.mountaingoatsoftware.com/blog/not-everything-needs-to-be-a-user-story-using-fdd-features



6 a. What are the key differences in characteristics between the Formal-incremental and Agile-iterative SDLC?

Formal

Explicit architecture
Explicit UX design
(end user consideration)
Explicit configuration

Agile

Productivity increase
Responsive to feedback
(client satisfaction)
Working software

b. How would these characteristics influence risk management?

Plan ahead

Plan Just-In-Time



Risk- bad examples

- “*Scope is ill-defined*”.
- “*The project may be late*”.
- “*Project estimates are very optimistic*”.
- “*Poor data quality*”.

Actual Poor Examples: (all taken from page 1 of Google in Dec 2015)



Risk- better examples

MELBOURCE

There is a risk that (think about how to complete)

- “*the export licence may not be granted.*”
- “*ground conditions may not be suitable for*”
- “*key (specific) system interfaces may not be compatible.*”
- “*there may not be the physical space for a required equipment.*”
- “*data rates for required image quality may exceed capacity.*”
- “*the regulator may introduce new requirements relating to...*”
- “*severe weather may impact progress*”
- “*(the requirement for) full spatial coverage may not be physically possible*”

Thank You!



IBM (2008): 40% of IT projects meet schedule, budget, & quality goals

<http://www-935.ibm.com/services/us/gbs/bus/>

pdf/gbe03100-user-03-making-change-work.pdf

KPMG (2013): a third of the IT spend for an organization delivers the desired results

<https://www.kpmg.com/NZ/en/IssuesAndInsights/ArticlesPublications/Documents/KPMG-Project-Management-Survey-2013.pdf>

XDNET (2009): estimate that the cost of failed IT projects are as high as \$6 Trillion worldwide

[http://www.zdnet.com/blog/projectfailures/
worldwide-cost-of-it-failure-6-2-trillion/7627](http://www.zdnet.com/blog/projectfailures/worldwide-cost-of-it-failure-6-2-trillion/7627)



SWEN90016

Software Processes & Project Management

Teams, Virtual Teams
Communication

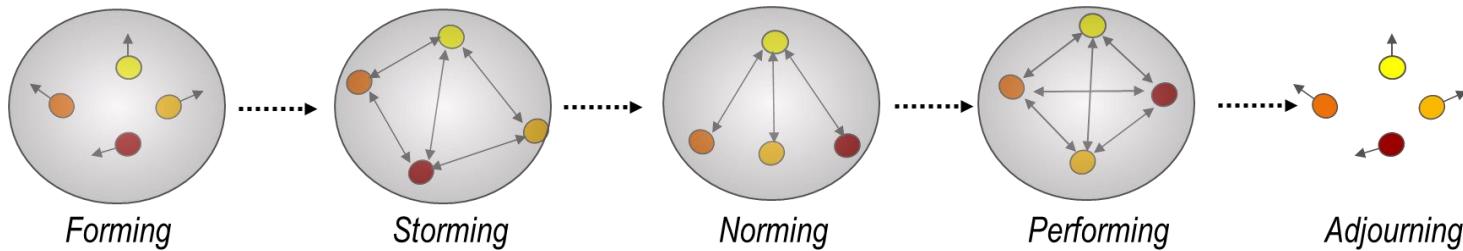


Explore **teams**- how teams are formed, and understand the different roles that individuals play in those teams.

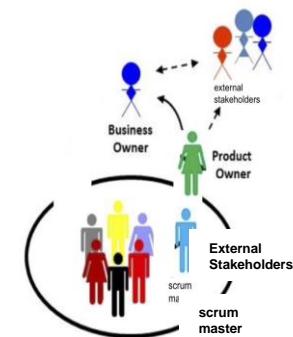
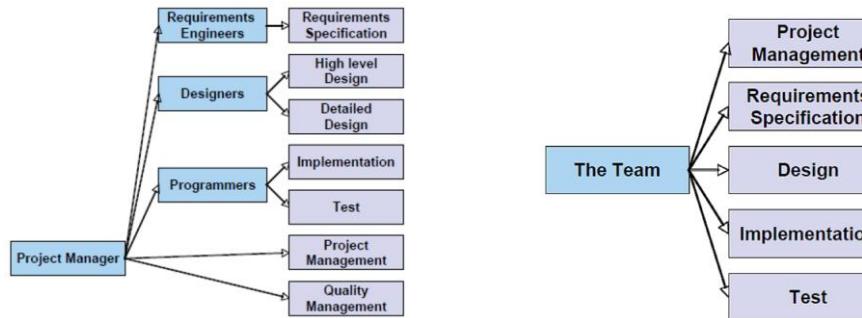
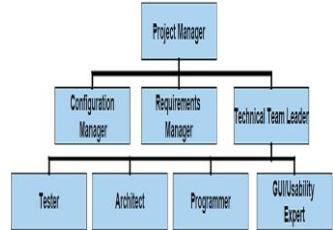


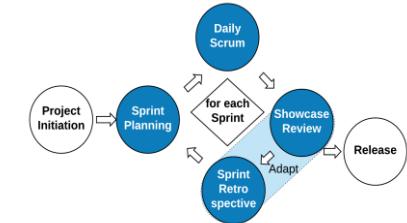
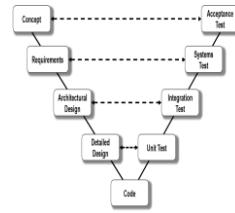
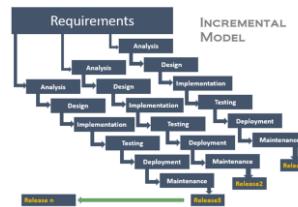
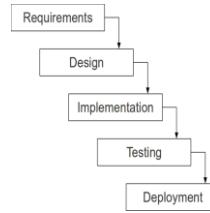


Team life cycle



Team structures





The relationships between team structures and lifecycle models constrains the selection of both.

The organisation structure and team structure will have an impact on the progress of the project.



Waterfall team

- typically, quite large - e.g. 15 people or more
- typically, specialist role types: PM, BA, developer, tester
- Project Manager
- Product Development team: business analysts understand the problem, document requirements
- Design team, data modeller(s), enterprise architect, designers
- Developers /programmers
- Software testing team
- Maintenance team – programmers
- Change management team: training program development and users

Agile team

- Face to face communication is crucial
- Feature driven development (FDD) rather than focusing on design documents
- ScrumMaster is responsible to provide team with the resources necessary
- Team members are self-organizing and self-select tasks in the Sprint Backlog
- Product Owner is responsible for assigning priorities to the requirements in a Product Backlog
- Extreme Programming Team: Programmers are paired



1. Consider the individuals in the team (10 mins)

Consider your own strengths and weaknesses

What **roles** might you play in your project team?

Initiator: offers ideas, solutions, brainstorm, lateral thinker

Information seeker: wants facts

Information giver: describes own experience, offers facts, clarification

Coordinator: combine contribution of others

Evaluator: assess quality of contributions

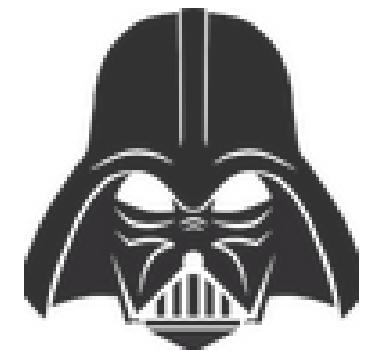
Encourager: praising, accepting, cohesion and warmth

Harmonizer: build consensus, humor to neutralize anger

Standard setter: focus on goals, standards

Follower: agreeable

Group observer: provides feedback





1. Consider the individuals in the team (10 mins)

Consider your own strengths and weaknesses

What **roles** might you play in your project team?

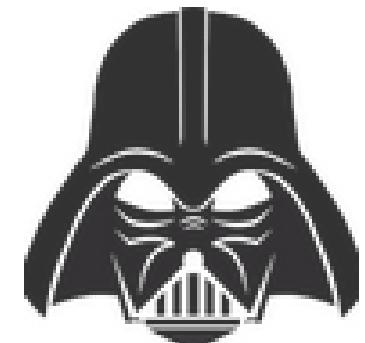
Effective team members who:

Are team focused rather than individually focussed

Are focused on content and processes

Can handle conflict

Are great communicators





What roles are necessary for a high functioning team?

Task

Initiator:

Information seeker:

Information giver:

Coordinator:

Evaluator:

Maintenance

Encourager:

Harmonizer:

Standard setter:

Follower:

Group observer:

Destructive

Blocker

Recognition seeker

Dominator:

Free rider:

Avoider:

Lone wolf:

Reasonable combination

Why Virtual Teams?

Why does it appeal to employees?

Employees can be more flexible with work and home commitments

Why does it appeal to organisations?

- Organisations can access the best GLOBAL talent
- Save on real estate costs



Explore **teams**- Communication.
Professional communication.
Virtual communication



What is a virtual team?

A **virtual team** (also known as a **geographically dispersed team, distributed team, or remote team**) usually refers to a group of individuals who work together from different geographic locations and rely on communication technology.¹

(Wikipedia)



How to communicate better

Wechat

Zoom

Whatsapp

Slack

What tools and technologies can help

Video on and off, wechat, body language- emojis



Thank You!



SWEN90016

Software Processes & Project Management

Cost Estimation



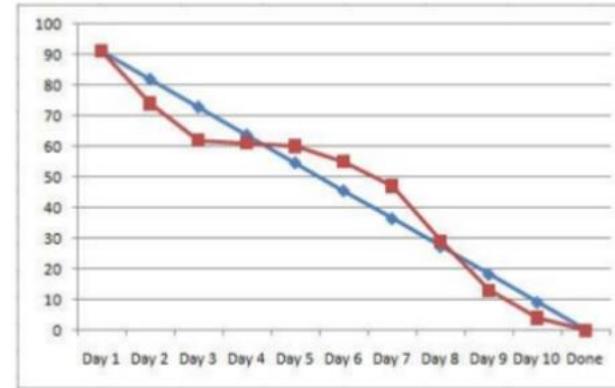
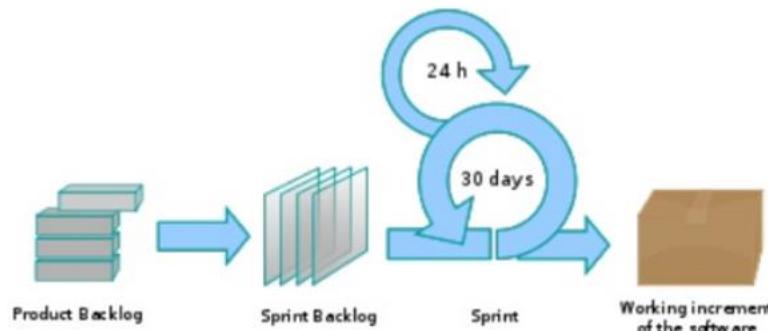
Become familiar with

Agile

User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II



Roles

Product Owner
Scrum Master
Development Team

Ceremonies

Daily Stand Up
Sprint Planning
Sprint Review
Sprint Retrospective

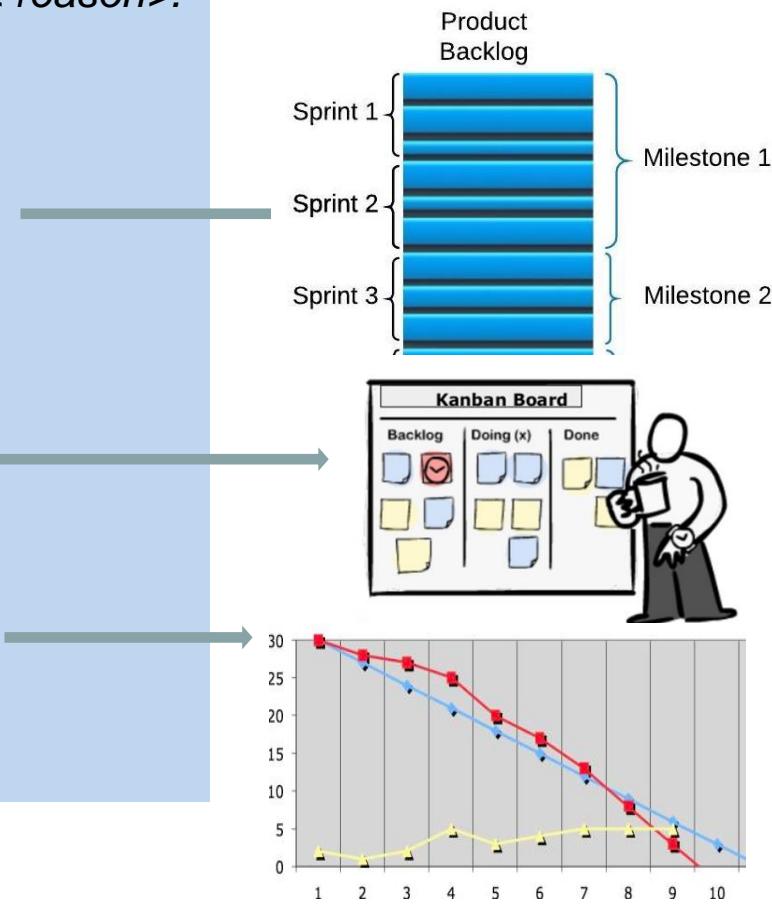
From Lecture 2, slide 51

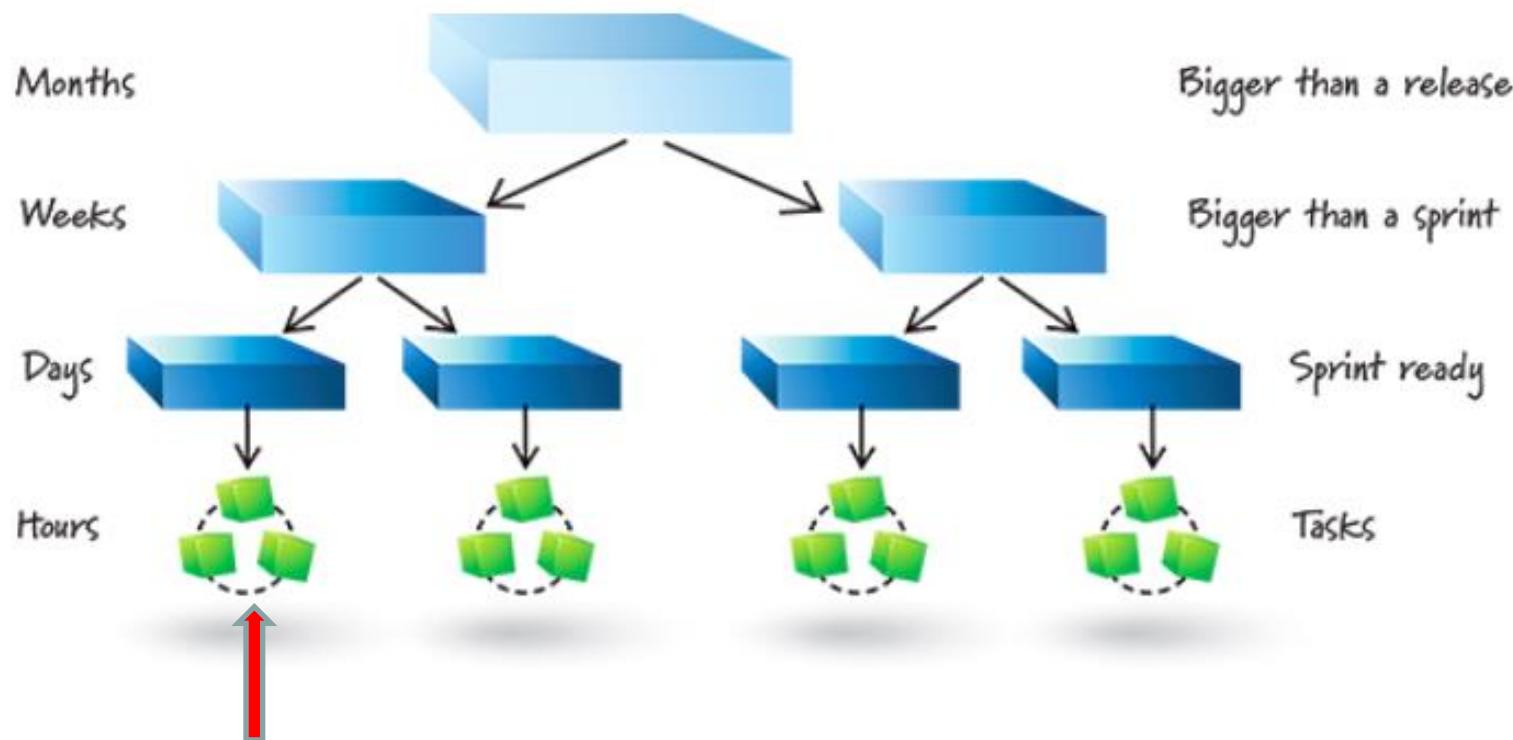
Artifacts

User Story
Product Backlog
Sprint Backlog
Burndown Chart
Burnup Chart



- User Stories
 - As a <user>, I want < goal> so that < reason>.
- Product Backlog
 - Features listed in client priority order
 - Release milestones annotated to list
- Sprint Backlog
 - Features selected for this iteration
 - Visual Kanban board
- Burn Down Chart
 - Measure the features **100% done**





Product Owner has a *conversation* with the Developer to understand requirement



(Sprint) User Story

- A developer's perspective
- A conversation placeholder

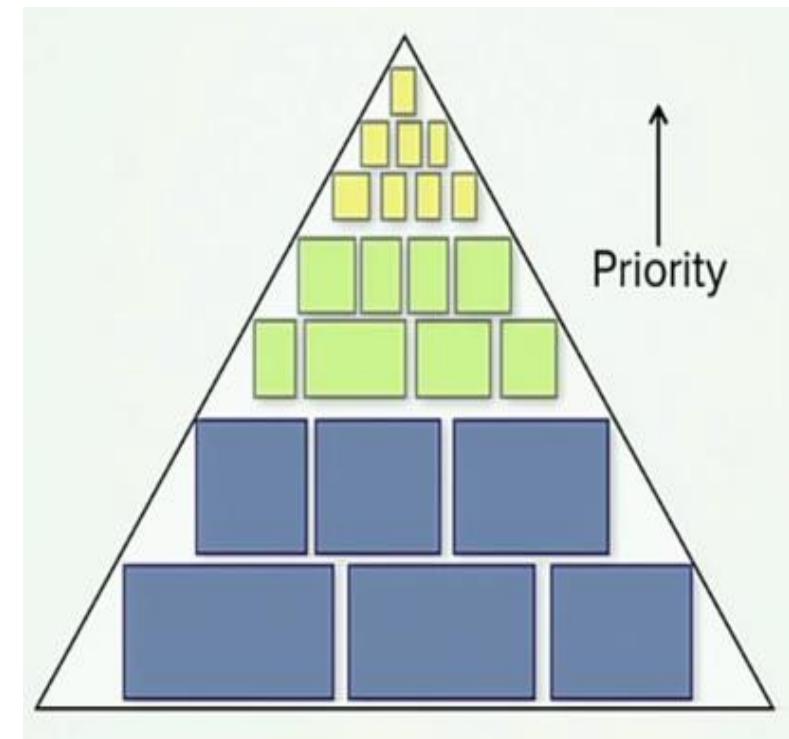
Feature User Story

- Product capabilities
- Product Owner perspective

Epic User Story

- New business services
- A product

Contentious! Advice from
the internet may vary ...





Story points: a relative measure of the size of a user story
(the requirements of the system are documented using user stories)

From Lecture 6, slide 71

raw values are unimportant

relative values matter

2 point story is twice as long
as a 1 point story

limit range of Sprint Backlog
estimates to 1-10





A practical Example of Size vs Duration

- I am tasked with moving a large pile of dirt from the front of my home to the back yard.
- I could look at the pile of dirt, assess my tools [a shovel and a wheelbarrow], and directly estimate the job at two hours.
- In arriving at this estimate I bypassed any estimate of size and went directly to an estimate of duration.



A practical Example of Size vs Duration

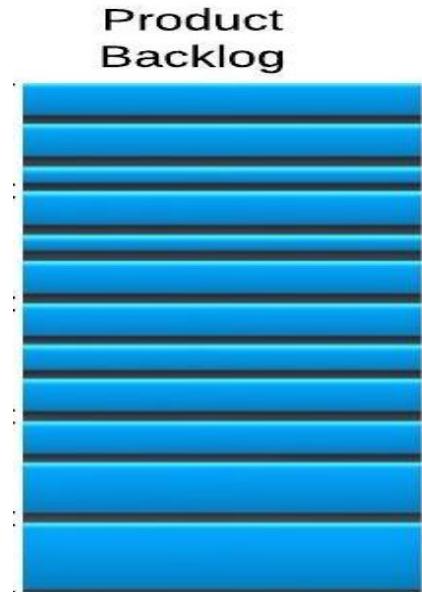
- Suppose instead that I look at the pile and estimate its size.
- Based on its dimensions I estimate the pile to contain about 100 cubic meters of dirt. This is my estimate of the size of this project.
- We want to know how long it will take to move the dirt: **Duration**
- We need to convert the estimate of size [100 cubic meters] into an estimate of duration.
- A label on my wheelbarrow says it has a capacity of two cubic meters.
- Dividing 100 cubic meters by 2 cubic meter, I decide that moving the dirt will take 50 trips with the wheelbarrow.
- I estimate that each trip will take three minutes to load the wheelbarrow, two minutes to walk to the back yard and dump the dirt, and one minute to walk back with the empty wheelbarrow. Total trip time will be six minutes.
- Since I anticipate making 50 trips taking 6 minutes each, my estimate of duration is 300 minutes or 5 hours.



Project:
phase
Initiation

Fixed Date and Time constraints

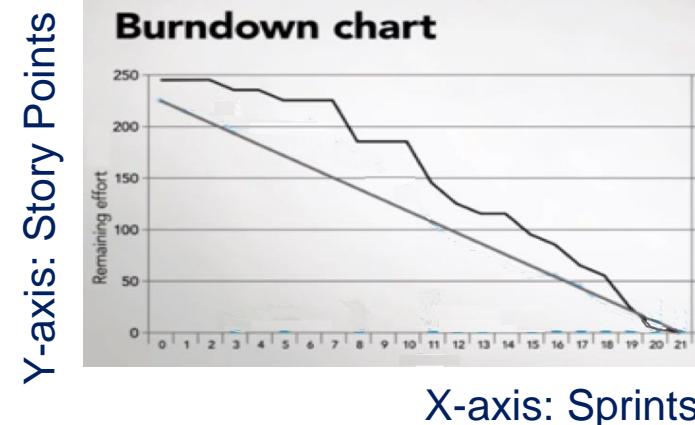
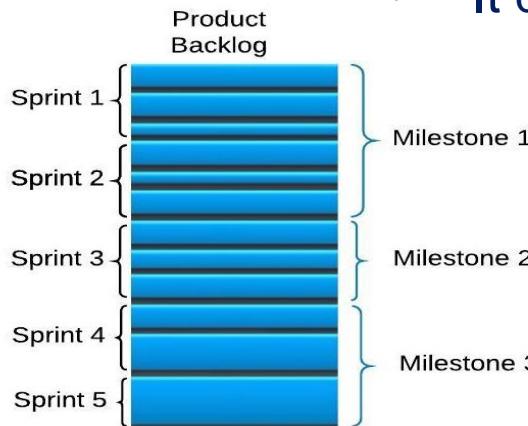
- Business Roadmap identifies candidate project
- Product vision established with external stakeholders
 - Create Product Backlog





Project phase:
Initial Sprint Planning

- The groomed Product Backlog is estimated in Story Points
 - Cheap & quick estimation
 - Low quality indicators of {easy, medium hard}
 - Let estimates have larger values, like 21 or 100 are valid
- Find the dev team's Story-Point **Velocity** measure
 - It determines the **release** schedule





Project phase:
every Sprint Planning

- Create Sprint Backlog Fixed Date and Time constraints
 - Select high value User Stories from Product Backlog
 - Use velocity to fit appropriate number of Story Points
- Decompose selected User Stories on Sprint Backlog
 - Do Just-In-Time detailed estimation
 - Check number of Story Points will still fit
 - Detailed high quality estimation
 - Let estimates have smaller values, like 1 or 10 are valid

Humans have good judgement across one order of magnitude, but beyond that, humans are unreliable



Project phase:
every Sprint Planning

Fixed Scope constraints

The User Stories can be decomposed into tasks,

- Optionally estimate tasks in hours

Less accurate www.scruminc.com/story-points-why-are-they-better-than/

- A full task level Sprint Backlog estimated in hours is equivalent to a formal schedule (Gantt)

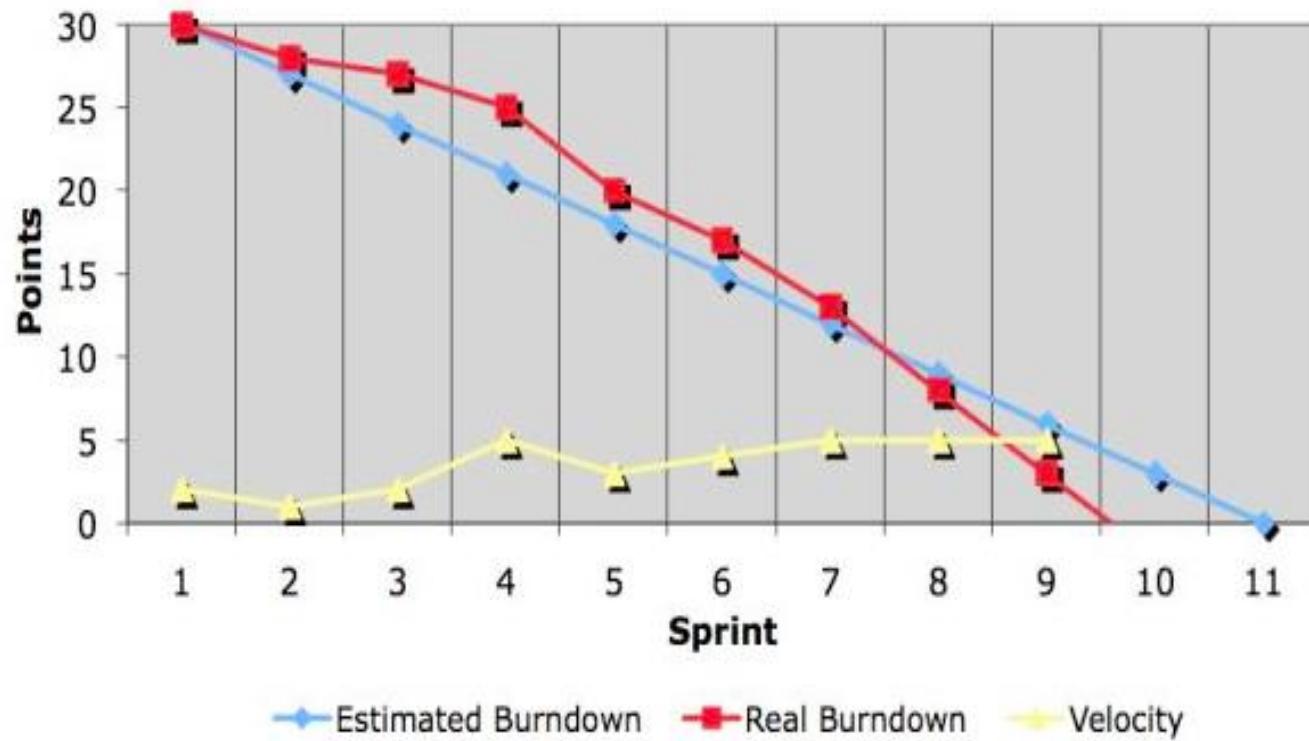
More work www.mountaingoatsoftware.com/agile/scrum/scrum-tools/sprint-backlog



Sprint Monitoring

Fixed Date and Time constraints

Project
phase:
Sprint



- Sprint Burn-down chart **monitors actual velocity**
- Scrum Master updated chart after daily standup

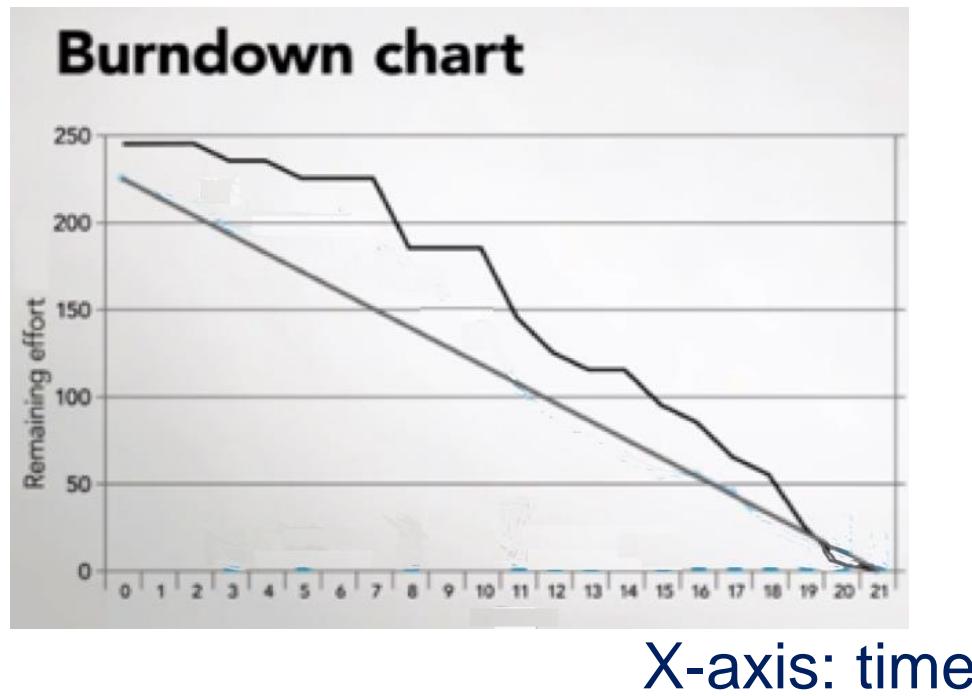


Fixed Date and Time constraints

Velocity determines the slope of the BurnDown charts

- The Scrum Master can track remaining effort
- Predict when the release milestones will be reached

Y-axis: effort



- Ideal schedule is the straight line
- Actual schedule is the jagged line
- The height of the chart shows the amount of work remaining



Burn Down Chart: Activity

A project has this groomed **Product Backlog**, consisting of these **User Stories** which have been estimated to have these **Story Points**.

An established development team has an average **velocity** of **seven** User Story Points per fortnight.

1. Estimate how many weeks this team will take to deliver?

$$\begin{aligned} \text{Total} &= \text{sum of SP} / \text{velocity per fortnight} \\ &= 35 \text{ total SP} / 7 \text{ velocity} \\ &= 5 \text{ fortnights} \end{aligned}$$

2. If the team actually completes the first two User Stories in two weeks, then what is the actual velocity of the team?

8 SP per fortnight, 4 per week

3. If a new User Story with Story Point=1 is added at the start of week 3, then in how many weeks do you estimate this project will take to be delivered now?

$$\begin{aligned} \text{Remaining} &= \text{sum of SP} / \text{velocity per fortnight} \\ &= (1+ 27) / 8 \\ &= 3.5 \text{ fortnights} \\ &= 7 \text{ weeks} \end{aligned}$$

$$\begin{aligned} \text{New Total} &= 7 + 2 \\ &= 9 \text{ weeks} \end{aligned}$$

Product Backlog	
User Story	Story Point
Story_1	3
Story_2	5
Story_3	13
Story_4	8
Story_5	1
Story_6	3
Story_7	2



Burn Down Chart: Activity

A project has this groomed **Product Backlog**, consisting of these **User Stories** which have been estimated to have these **Story Points**.

An established development team has an average **velocity** of **seven** User Story Points per fortnight.

Product Backlog	
User Story	Story Point
Story_1	3
Story_2	5
Story_3	13
Story_4	8
Story_5	1
Story_6	3
Story_7	2

4. Will User_Story_3 fit into a single sprint?

NO

5. What process does Scrum have for completing User_Story_3 ?

Break it down to smaller user story(s)



Top Down strategy:

Use cost of a previous similar project, size and effort
Source Lines of Code, Function Points, Cocomo

Bottom up strategy:

Estimate individual work items and sum
WBS, Agile Story Points and Velocity

Parametric:

use project characteristics in a mathematical model
NVP, ROI, IRR



Become familiar with

Agile

User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II

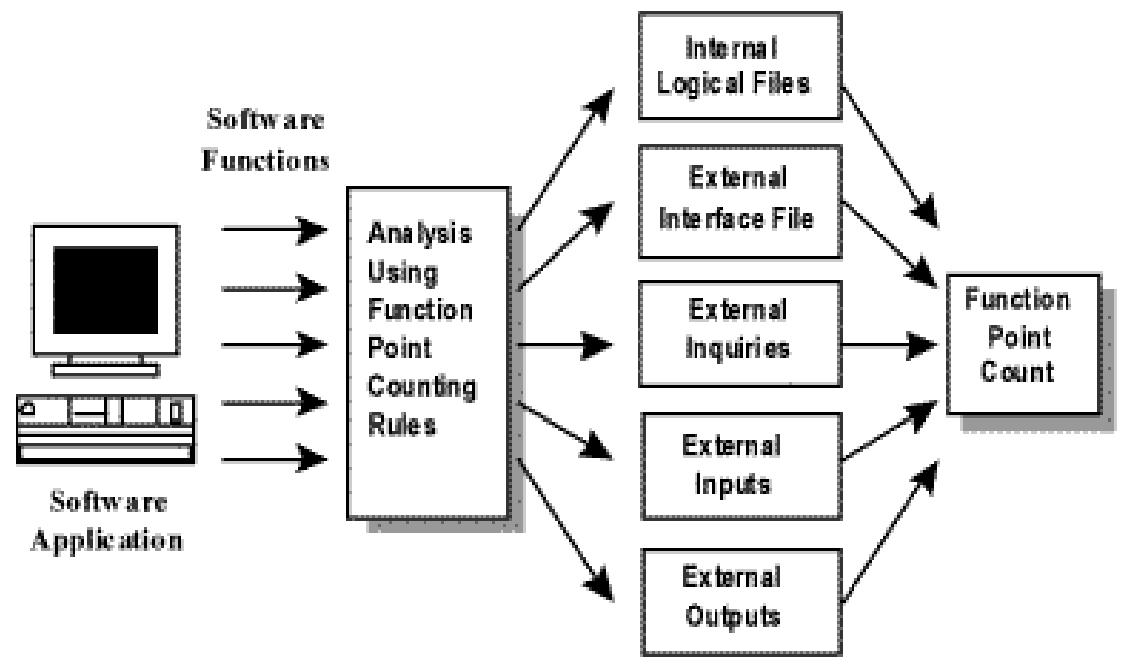


PMBOK

Historic Data

Done at any
time in project
lifecycle

What are they?





FP Computation Steps

1. Categorize functional requirements and count

Example: *Category* = {internal file, external file, input, output, query}

2. Estimate a *Complexity Level* for each category

Complexity Level = {simple, average, complex}

3. Compute *count total* of Function Points,
(see next slide)

Unadjusted Function Points =
sum (functions * complexity value)

4. Estimate *Value Adjustment Factors*

Value Adjustment Factor =
apply expert opinion to your project estimates

5. Compute *total function point count*

Adjusted Function Points =
multiply business function by VAF



FP Computation Steps

1. Categorize functional requirements and count

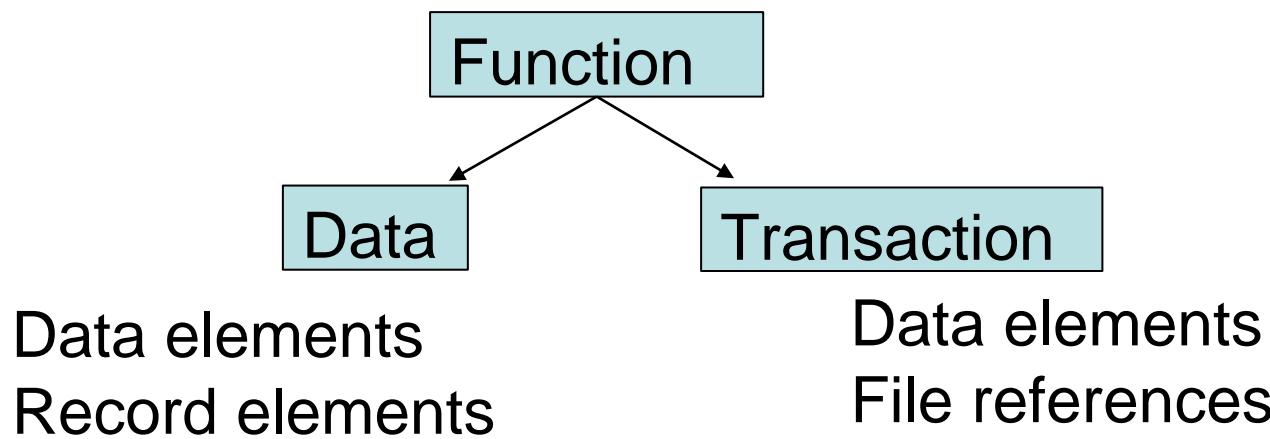


Example: *Category* = {internal file, external file, input, output, query}

2. Estimate a
Complexity Level
for each category

Complexity Level = {simple, average, complex}

Count functions from the Software Requirements Specification (SRS)





Historic Data

complexity values

Category	Simple Function Count	Weight	Average Function Count	Weight	Complex Function Count	Weight	Sub total
Internal Logical File	5	3		4	2	6	
External Interface File`		4		5	1	7	
External Input	2	3		4		6	
External Output	5	7	2	10	2	15	
External Inquiries/Queries	2	5		7		10	
Unadjusted Total							

Factors published from 2,192 recent Function Point projects

<http://www.qsm.com/resources/function-point-languages-table>



Given the following business functions,
how many *Unadjusted* Function Points exist?

Fill in the table.

Category	Simple Function Count	Weight	Average Function Count	Weight	Complex Function Count	Weight	Sub total
Internal Logical File	5	3		4	2	6	
External Interface File`		4		5	1	7	
External Input	2	3		4		6	
External Output	5	7	2	10	2	15	
External Inquiries/Queries	2	5		7		10	
Unadjusted Total							



Step 3: Calculate Functional Points

Category	Simple Function Count	Weight	Average Function Count	Weight	Complex Function Count	Weight	Sub total
Internal Logical File	5	3		4	2	6	27
External Interface File`		4		5	1	7	7
External Input	2	3		4		6	6
External Output	5	7	2	10	2	15	85
External Inquiries/Queries	2	5		7		10	10
Unadjusted Total							135



Historic Data

Give the 14 system characteristics, estimate how relevant they are to your system, use the ***typical weights***

0 = no effect

1 = incidental

2 = moderate

3 = average

4 = significant

5 = essential

Total VAF = 40

TABLE 6-2 Function Point System Characteristics

System Characteristic	
Data communications required	2
Distributed processing	1
Performance needs	5
Heavily utilized operating environment	4
On-line data entry	4
Backup and recovery	4
Master file access online	3
Transaction input complexity	2
Internal processing complexity	2
Reusable code	2
Input, outputs, files, inquiries complex	2
Designed for multiple sites	4
Designed to facilitate change	3
Installation complexity	2
Total	40



Compute **Adjusted Function Points** using formula:
Unadjusted FP * (0.65 + 0.01 * VAF)

$$= 135 * (0.65 + 0.01 * 40)$$

$$= 135 * (0.65 + 0.40)$$

$$= 135 * (1.05)$$

$$= 141.75 \text{ Adjusted Functional Points}$$





The Constructive Cost Model:

Here is a playpen to try: <http://csse.usc.edu/tools/cocomoii.php>

Fill in the details for the Language Research Project.

Extra details to get started: let there be:

Sizing method: **135 Function Points**

The **Java** development language

The **cost per person-month** is **\$1500**



Thank You!



SWEN90016

Software Processes & Project Management

Cost Estimation



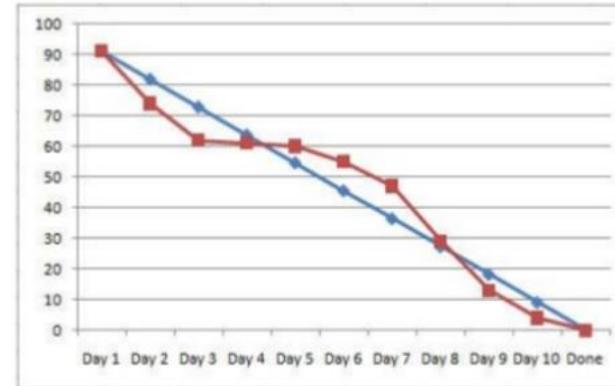
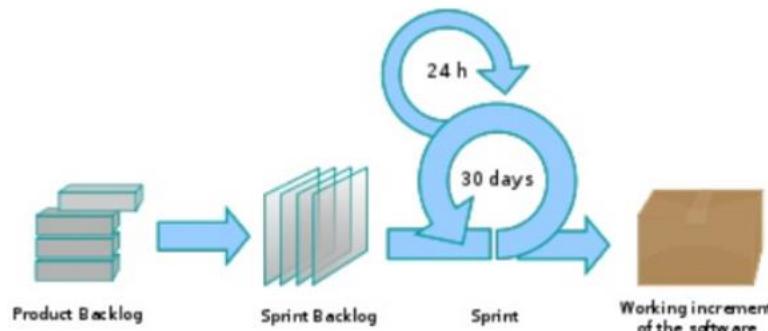
Become familiar with

Agile

User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II



Roles

Product Owner
Scrum Master
Development Team

Ceremonies

Daily Stand Up
Sprint Planning
Sprint Review
Sprint Retrospective

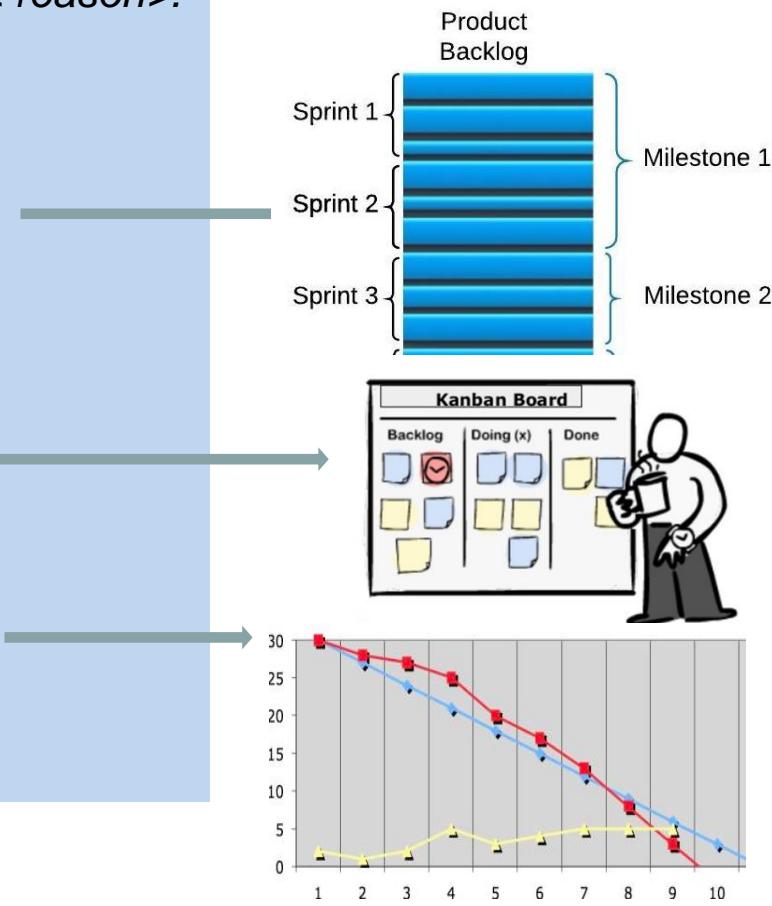
From Lecture 2, slide 51

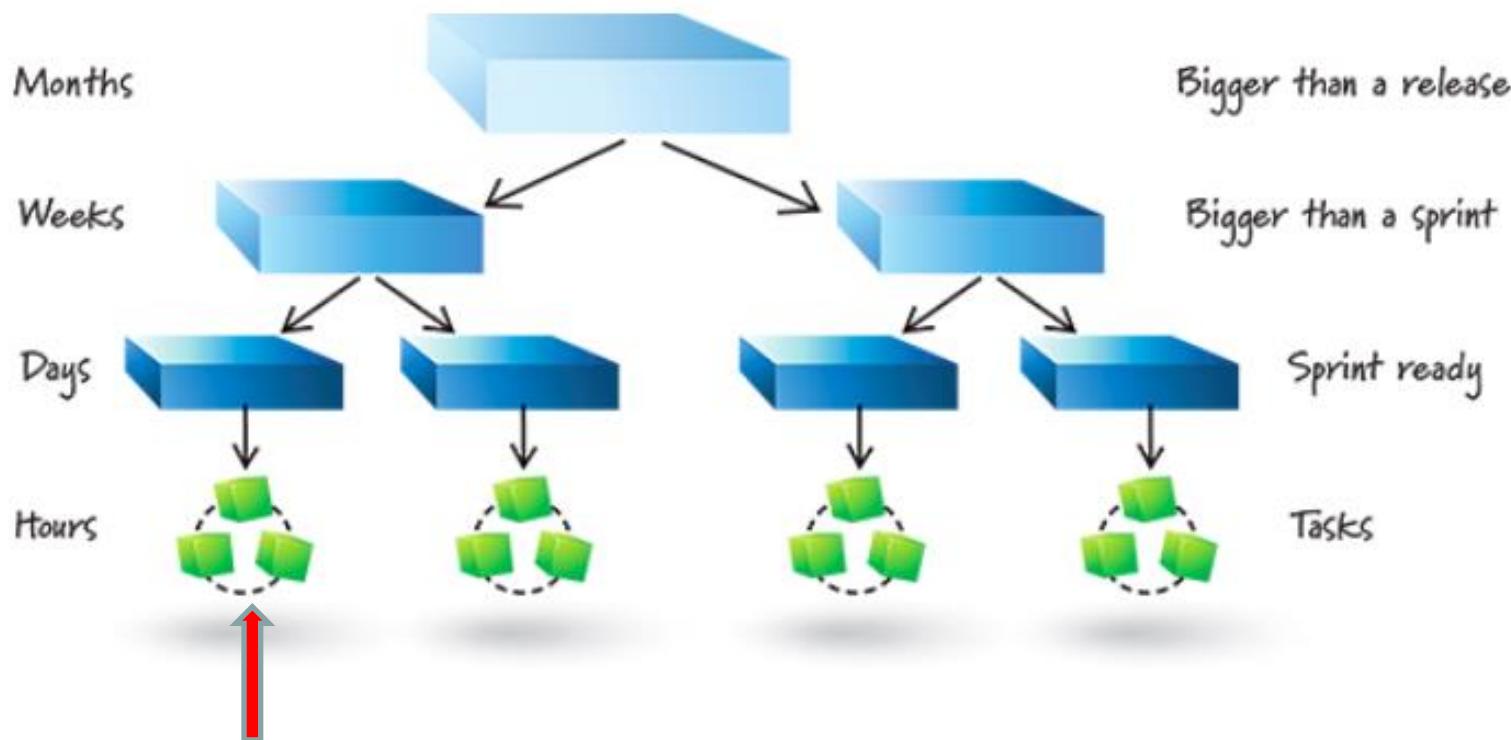
Artifacts

User Story
Product Backlog
Sprint Backlog
Burndown Chart
Burnup Chart



- User Stories
 - As a <user>, I want < goal> so that < reason>.
- Product Backlog
 - Features listed in client priority order
 - Release milestones annotated to list
- Sprint Backlog
 - Features selected for this iteration
 - Visual Kanban board
- Burn Down Chart
 - Measure the features **100% done**





Product Owner has a *conversation* with the Developer to understand requirement



(Sprint) User Story

- A developer's perspective
- A conversation placeholder

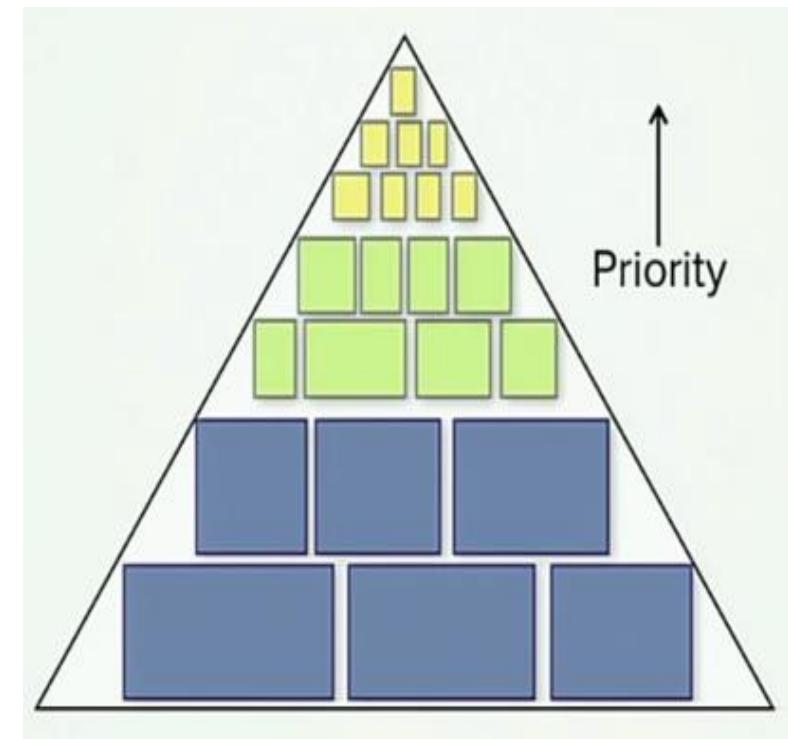
Feature User Story

- Product capabilities
- Product Owner perspective

Epic User Story

- New business services
- A product

Contentious! Advice from the internet may vary ...





Story points: a relative measure of the size of a user story
(the requirements of the system are documented using user stories)

From Lecture 6, slide 71

raw values are unimportant

relative values matter

2 point story is twice as long
as a 1 point story

limit range of Sprint Backlog
estimates to 1-10





A practical Example of Size vs Duration

- I am tasked with moving a large pile of dirt from the front of my home to the back yard.
- I could look at the pile of dirt, assess my tools [a shovel and a wheelbarrow], and directly estimate the job at two hours.
- In arriving at this estimate I bypassed any estimate of size and went directly to an estimate of duration.



A practical Example of Size vs Duration

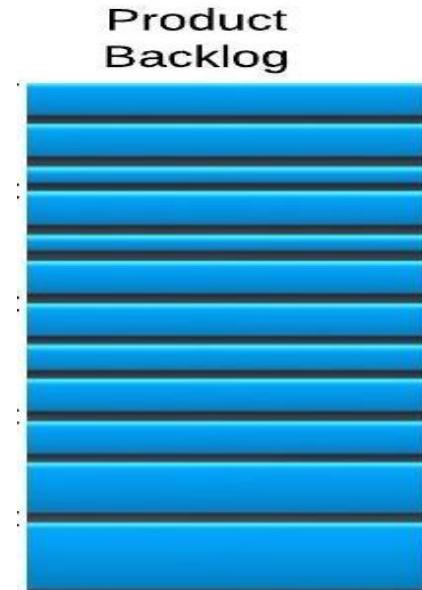
- Suppose instead that I look at the pile and estimate its size.
- Based on its dimensions I estimate the pile to contain about 100 cubic meters of dirt. This is my estimate of the size of this project.
- We want to know how long it will take to move the dirt: **Duration**
- We need to convert the estimate of size [100 cubic meters] into an estimate of duration.
- A label on my wheelbarrow says it has a capacity of two cubic meters.
- Dividing 100 cubic meters by 2 cubic meter, I decide that moving the dirt will take 50 trips with the wheelbarrow.
- I estimate that each trip will take three minutes to load the wheelbarrow, two minutes to walk to the back yard and dump the dirt, and one minute to walk back with the empty wheelbarrow. Total trip time will be six minutes.
- Since I anticipate making 50 trips taking 6 minutes each, my estimate of duration is 300 minutes or 5 hours.



Project:
phase
Initiation

Fixed Date and Time constraints

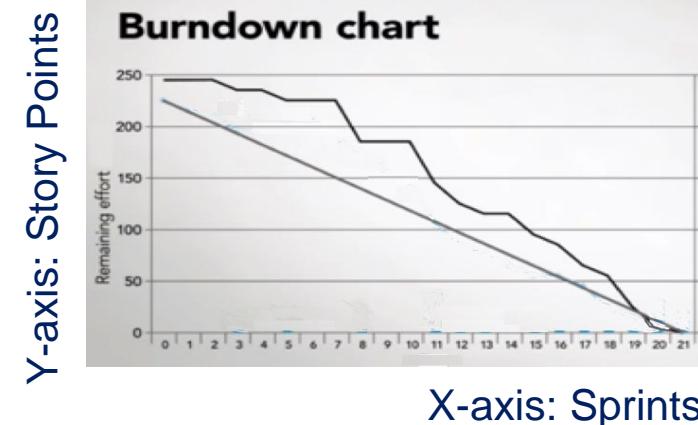
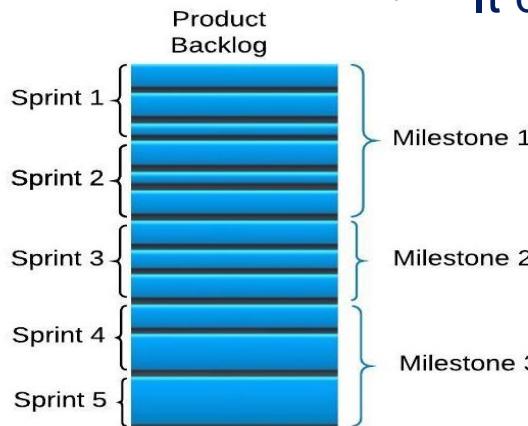
- Business Roadmap identifies candidate project
- Product vision established with external stakeholders
 - Create Product Backlog





Project phase:
Initial Sprint Planning

- The groomed Product Backlog is estimated in Story Points
 - Cheap & quick estimation
 - Low quality indicators of {easy, medium hard}
 - Let estimates have larger values, like 21 or 100 are valid
- Find the dev team's Story-Point **Velocity** measure
 - It determines the **release** schedule





Project phase:
every Sprint Planning

- Create Sprint Backlog Fixed Date and Time constraints
 - Select high value User Stories from Product Backlog
 - Use velocity to fit appropriate number of Story Points
- Decompose selected User Stories on Sprint Backlog
 - Do Just-In-Time detailed estimation
 - Check number of Story Points will still fit
 - Detailed high quality estimation
 - Let estimates have smaller values, like 1 or 10 are valid

Humans have good judgement across one order of magnitude, but beyond that, humans are unreliable



Project phase:
every Sprint Planning

Fixed Scope constraints

The User Stories can be decomposed into tasks,

- Optionally estimate tasks in hours

Less accurate www.scruminc.com/story-points-why-are-they-better-than/

- A full task level Sprint Backlog estimated in hours is equivalent to a formal schedule (Gantt)

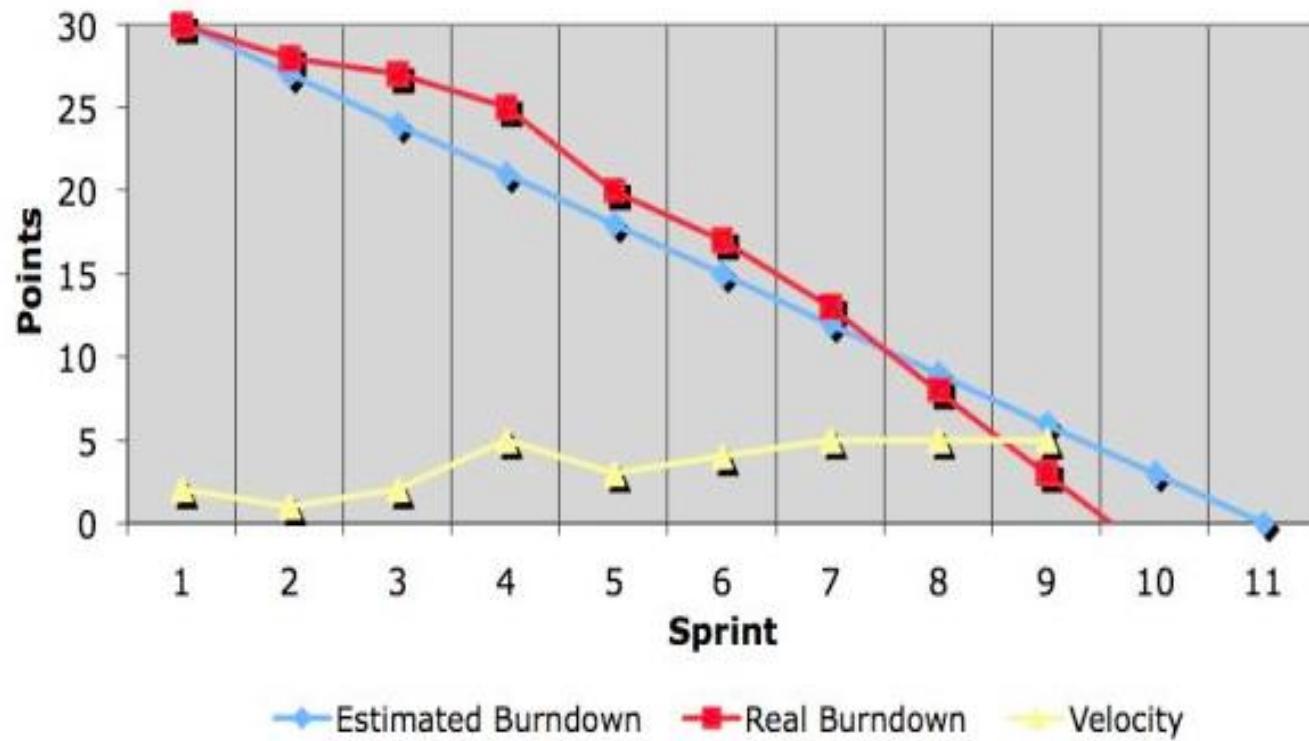
More work www.mountaingoatsoftware.com/agile/scrum/scrum-tools/sprint-backlog



Sprint Monitoring

Fixed Date and Time constraints

Project
phase:
Sprint



- Sprint Burn-down chart **monitors actual velocity**
- Scrum Master updated chart after daily standup

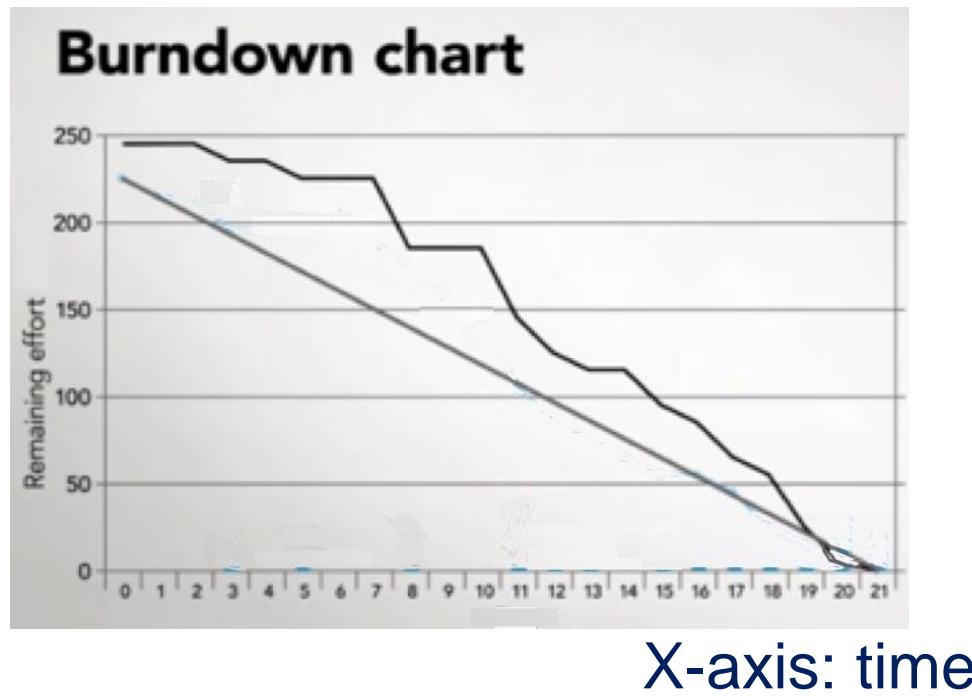


Fixed Date and Time constraints

Velocity determines the slope of the BurnDown charts

- The Scrum Master can track remaining effort
- Predict when the release milestones will be reached

Y-axis: effort



- Ideal schedule is the straight line
- Actual schedule is the jagged line
- The height of the chart shows the amount of work remaining



Burn Down Chart: Activity

A project has this groomed **Product Backlog**, consisting of these **User Stories** which have been estimated to have these **Story Points**.

An established development team has an average **velocity** of **seven** User Story Points per fortnight.

Product Backlog	
User Story	Story Point
Story_1	3
Story_2	5
Story_3	13
Story_4	8
Story_5	1
Story_6	3
Story_7	2

1. Estimate how many weeks this team will take to deliver?
2. If the team actually completes the first two User Stories in two weeks, then what is the actual velocity of the team?
3. If a new User Story with Story Point=1 is added at the start of week 3, then in how many weeks do you estimate this project will take to be delivered now?



Burn Down Chart: Activity

A project has this groomed **Product Backlog**, consisting of these **User Stories** which have been estimated to have these **Story Points**.

An established development team has an average **velocity** of **seven** User Story Points per fortnight.

Product Backlog	
User Story	Story Point
Story_1	3
Story_2	5
Story_3	13
Story_4	8
Story_5	1
Story_6	3
Story_7	2

4. Will User_Story_3 fit into a single sprint?
5. What process does Scrum have for completing User_Story_3 ?



Top Down strategy:

Use cost of a previous similar project, size and effort
Source Lines of Code, Function Points, Cocomo

Bottom up strategy:

Estimate individual work items and sum
WBS, Agile Story Points and Velocity

Parametric:

use project characteristics in a mathematical model
NVP, ROI, IRR



Become familiar with

Agile

User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II

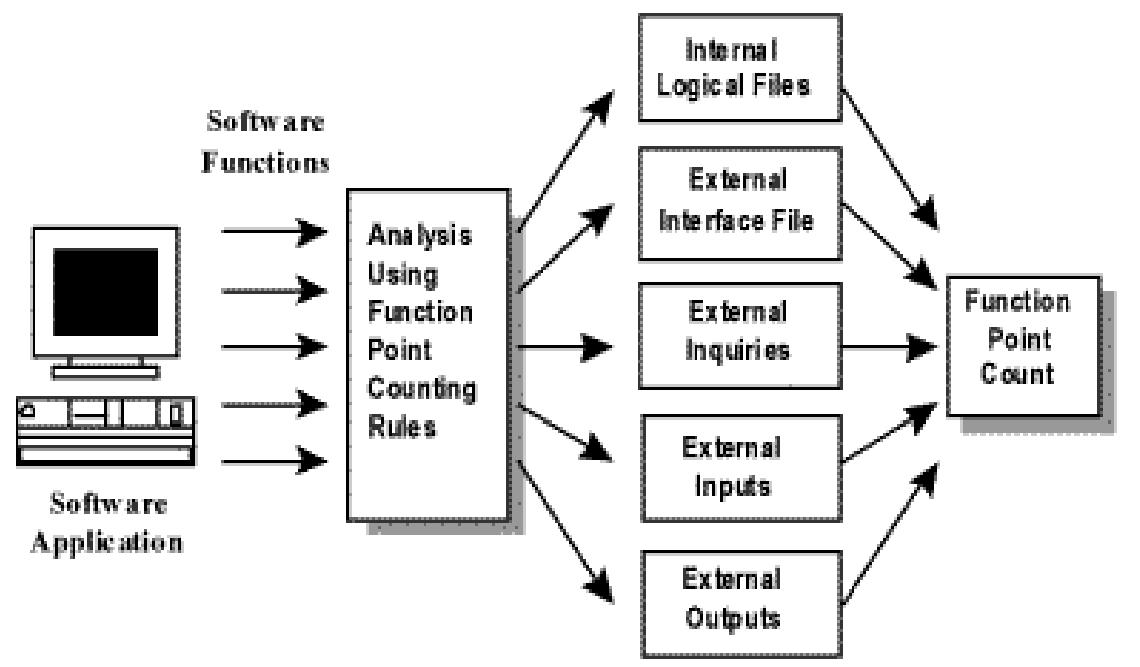


PMBOK

Historic Data

Done at any
time in project
lifecycle

What are they?





FP Computation Steps

1. Categorize functional requirements and count

Example: *Category* = {internal file, external file, input, output, query}

2. Estimate a *Complexity Level* for each category

Complexity Level = {simple, average, complex}

3. Compute *count total* of Function Points,
(see next slide)

Unadjusted Function Points =
sum (functions * complexity value)

4. Estimate *Value Adjustment Factors*

Value Adjustment Factor =
apply expert opinion to your project estimates

5. Compute *total function point count*

Adjusted Function Points =
multiply business function by VAF



FP Computation Steps

1. Categorize functional requirements and count

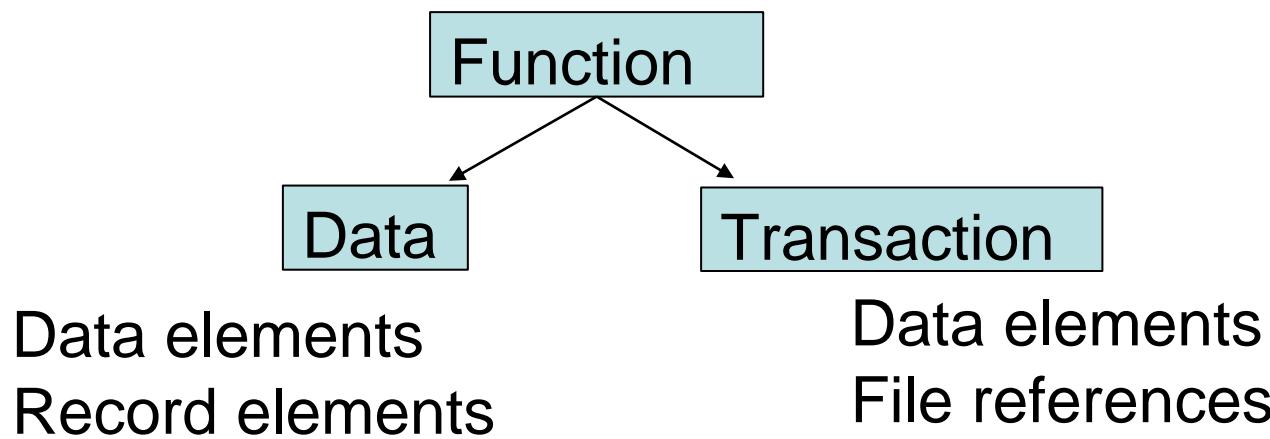


Example: *Category* = {internal file, external file, input, output, query}

2. Estimate a
Complexity Level
for each category

Complexity Level = {simple, average, complex}

Count functions from the Software Requirements Specification (SRS)





Historic Data

complexity values

Category	Simple Function Count	Weight	Average Function Count	Weight	Complex Function Count	Weight	Sub total
Internal Logical File	5	3		4	2	6	
External Interface File`		4		5	1	7	
External Input	2	3		4		6	
External Output	5	7	2	10	2	15	
External Inquiries/Queries	2	5		7		10	
Unadjusted Total							

Factors published from 2,192 recent Function Point projects

<http://www.qsm.com/resources/function-point-languages-table>



Given the following business functions,
how many *Unadjusted* Function Points exist?

Fill in the table.

Category	Simple Function Count	Weight	Average Function Count	Weight	Complex Function Count	Weight	Sub total
Internal Logical File	5	3		4	2	6	
External Interface File`		4		5	1	7	
External Input	2	3		4		6	
External Output	5	7	2	10	2	15	
External Inquiries/Queries	2	5		7		10	
Unadjusted Total							



Historic Data

Give the 14 system characteristics, estimate how relevant they are to your system, use the ***typical weights***

0 = no effect

1 = incidental

2 = moderate

3 = average

4 = significant

5 = essential

TABLE 6-2 Function Point System Characteristics

System Characteristic	
Data communications required	2
Distributed processing	1
Performance needs	5
Heavily utilized operating environment	4
On-line data entry	4
Backup and recovery	4
Master file access online	3
Transaction input complexity	2
Internal processing complexity	2
Reusable code	2
Input, outputs, files, inquiries complex	2
Designed for multiple sites	4
Designed to facilitate change	3
Installation complexity	2
Total	40



Step 5: Calculate Adjusted FP

Compute **Adjusted Function Points** using formula:
Unadjusted FP * (0.65 + 0.01 * VAF)





The Constructive Cost Model:

Here is a playpen to try:

<http://softwarecost.org/tools/COCOMO/>

Fill in the details for the Language Research Project.

Extra details to get started: let there be:

Sizing method: **135 Function Points**

The **Java** development language

The **cost per person-month** is **\$1500**



Thank You!



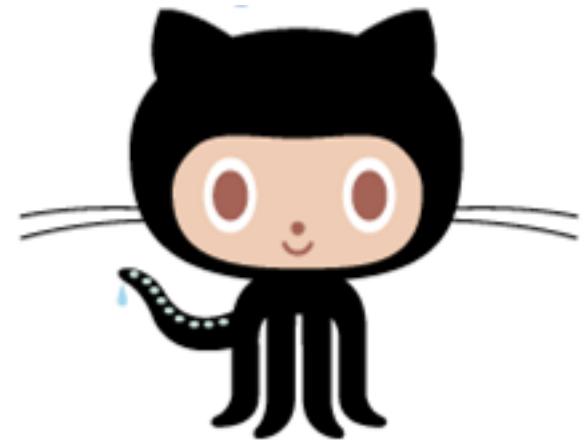
SWEN90016

Software Processes & Project Management

Configuration
and
Version Control



Become familiar with
Continuous Integration
&
GIT



<https://github.com/microsoft>



The Version Control System Continuous Integration (CI)

(not be confused with CI- configuration items –lecture)

- Practice of automating the integration of code changes from multiple contributors into a single software project.
- [DevOps best practice](#), developers merge code changes into a central repository where builds and tests then run.
 - Automated tools are used to assert the new code's correctness before integration.



- Version Control System

who, what, where, when of code

- Made by Linus Torvalds!

famous guru - transformed technology twice

Created kernel for Linux OSs , Android and Chrome OS

Created Git, the source code management system



meet Tux, slightly overweight penguin



From Lecture 10, slide 14

- **WHAT:** identify artefacts under versioning control
- **WHERE:** items stored in REPO: version control repository
- **WHO:** tags items with your name (collection remains consistent)
- **WHEN:** tags item with date (versions exist)
- **WHY:** quality code sharing, back up, roll-back, parallel branching

The “why” is recorded by your comment on the commit!



- Distributed (everyone has their own code repository local to them!)
- Open Source (everyone likes open source code 😊)
- Bomb proof

GitHub hit by Massive DDoS Attack From China

Friday, March 27, 2015 by Mohit Kumar

[Tweet](#) [G+](#) [Share](#) [Share](#) 29 [in Share](#) [f Share](#) [Share](#)





1.git init

Initialize local repository called **master**

1.git add *filename*

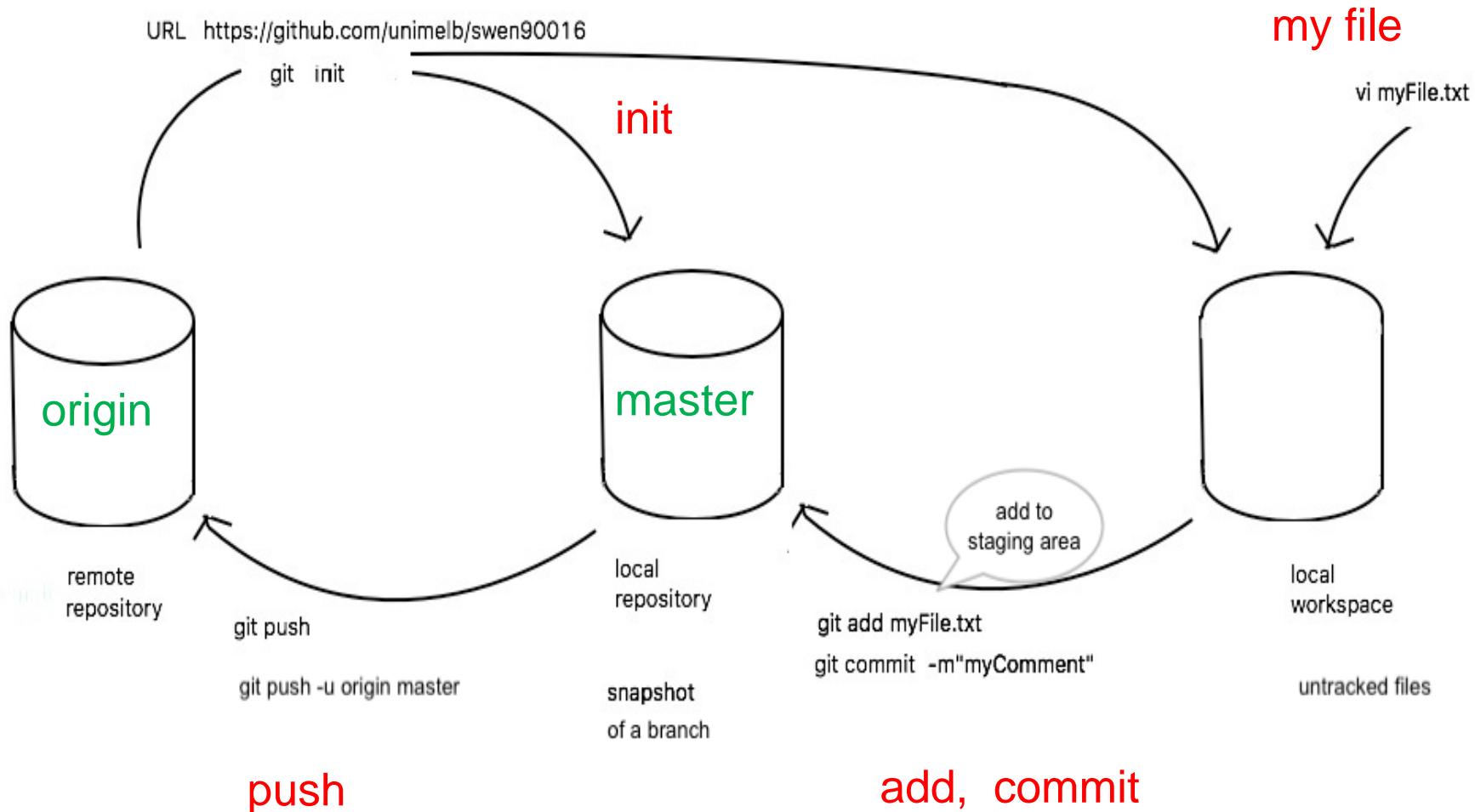
Track changes made to the contents
of this file

1.git commit -m “*message*”

Save this file in local repository

1.git push

Save this collection
- to remote repository called **origin**
- from local repository called **master**





No Coding Necessary

To complete this tutorial, you need a

<https://github.com/microsoft> and Internet access. You don't need to know how to code, use the command line, or install Git (the version control software GitHub is built on).

Tip: Open this guide in a separate browser window (or tab) so you can see it while you complete the steps in the tutorial.



Step 1. Create a Repository

A **repository** is usually used to organise a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.

GitHub makes it easy to add one at the same time you create your new repository. *It also offers other common options such as a license file.*

Your hello-world repository can be a place where you store ideas, resources, or even share and discuss things with others.



To Create a New Repository

In the upper right corner, next to your avatar or identicon, click + and then select **New repository**.

Name your repository *hello-world*.

Write a short description.

Select **Initialize this repository with a README**.

Owner Repository name

PUBLIC Great repository names are short and memorable. Need inspiration? How about [petulant-shame](#).

Description (optional)

Public Anyone can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

Initialize this repository with a README This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: Add a license:

Create repository

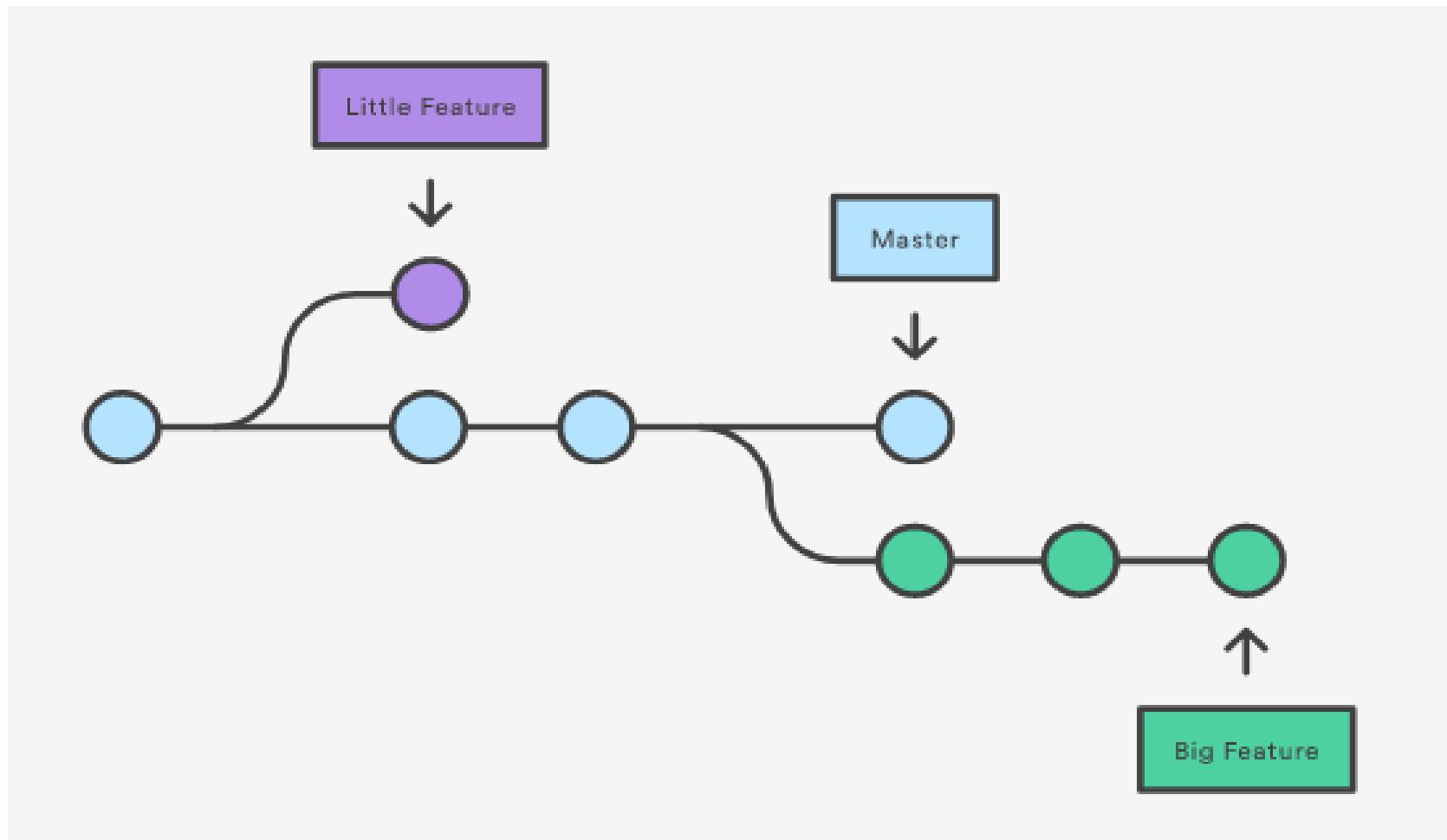


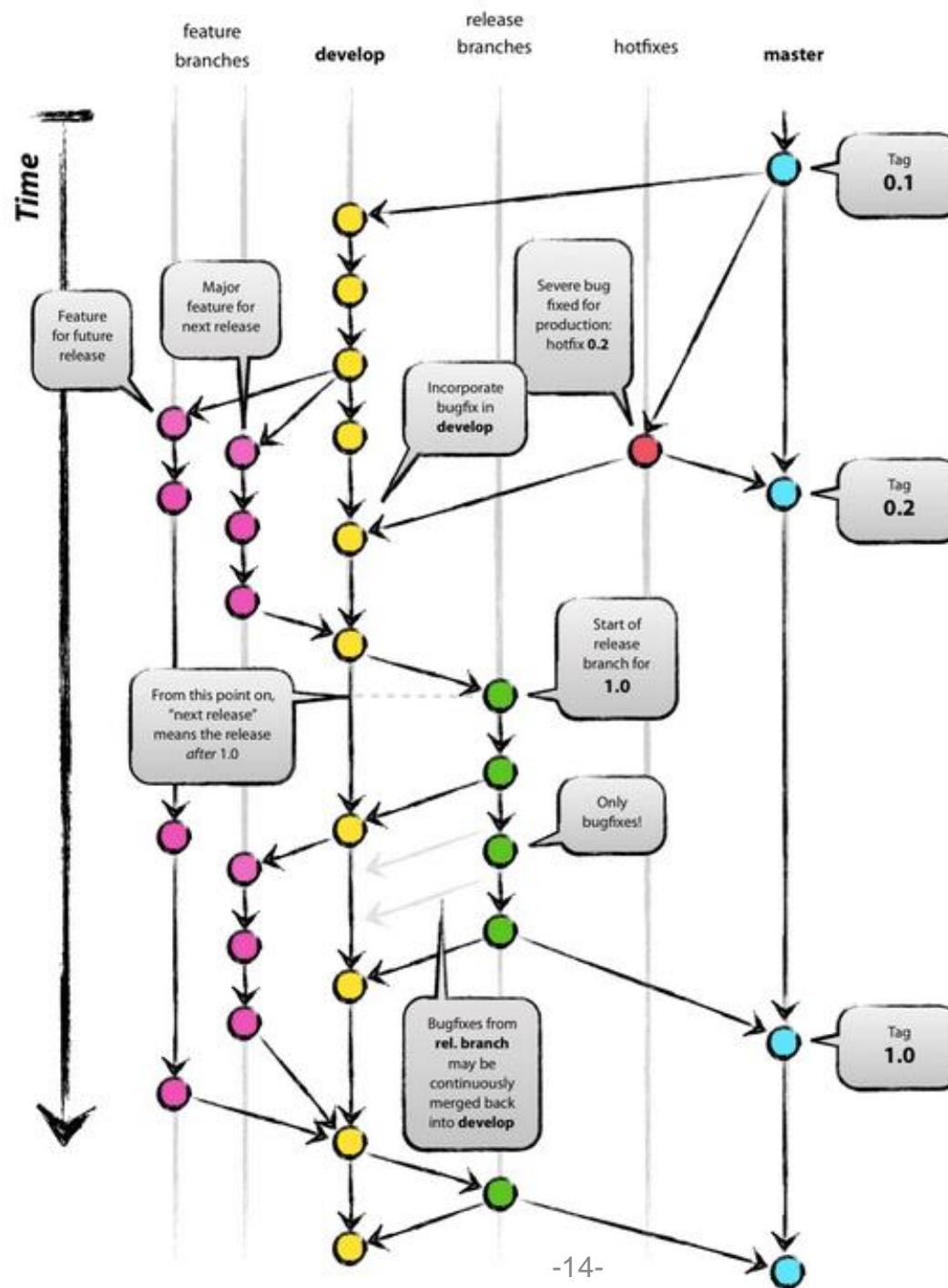
Step 2. Create a Branch

Branching is the way to work on different versions of a repository at one time.

By default, your repository has one branch named master which is considered to be the definitive branch. We use branches to experiment and make edits before committing them to master.

When you create a branch off the master branch, you're making a copy, or snapshot, of master as it was at that point in time. If someone else made changes to the master branch while you were working on your branch, you could pull in those updates.







Step 2. Create a Branch

This diagram shows:

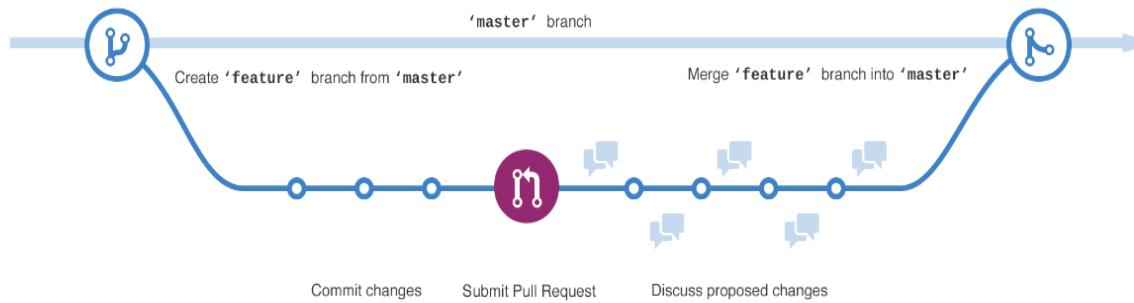
The master branch

A new branch called feature (because we're doing 'feature work' on this branch)

The journey that feature takes before it's merged into master

Have you ever saved different versions of a file? Something like:

story.txt
story-joe-edit.txt





To Create a New Branch

1. Go to your new repository hello-world.
2. Click the drop down at the top of the file list that says **branch: master**.
3. Type a branch name, readme-edits, into the new branch text box.
4. Select the blue **Create branch** box or hit “Enter” on your keyboard.
5. Now you have two branches, master and readme-edits. They look exactly the same, but not for long! Next, we’ll add our changes to the new branch. `story-joe-edit-reviewed.txt`



Just another repository — Edit

1 commit 1 branch

branch: master / hello-world / +

Initial README.Rmd Rmd

Switch branches/tags

readme-edit

Branches Tags

Create branch: readme-edits from 'master'

The screenshot shows a GitHub repository named "hello-world". The main page displays 1 commit and 1 branch. A dropdown menu is open under the "branch" button, showing the current branch "master" and a new branch "readme-edit" which is highlighted with a blue border. Below the dropdown, there are tabs for "Branches" and "Tags". At the bottom of the dropdown, a large blue button with white text says "Create branch: readme-edits from 'master'". On the left side of the page, there are file icons for "Initial README.Rmd" and "Rmd".

Now you have two branches, `master` and `readme-edits`. They look exactly the same, but not for long! Next we'll add our changes to the new branch.



Step 3. Make and Commit Changes

Bravo! Now, you're on the code view for your *readme-edits* branch, which is a copy of *master*. Let's make some edits.

On GitHub, saved changes are called *commits*. Each commit has an associated *commit message*, which is a description explaining why a particular change was made. Commit messages capture the history of your changes, so other contributors can understand what you've done and why.



Step 3. Make and Commit Changes

Click the README.md file.

Click the pencil icon in the upper right corner of the file view to edit.

In the editor, write a bit about yourself.

Write a commit message that describes your changes.

Click **Commit changes** button.

These changes will be made to just the README file on your *readme-edits* branch,
so now this branch contains content that's different from *master*.



The screenshot shows a GitHub repository page for 'hubot / hello-world'. The main area displays the 'README.md' file content:

```
1 # hello-world
2
3 Hi Humans!
4
5 Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
6 I've had tacos on the moon and find them far superior to Earth tacos.
7
```

Below the code editor, there is a 'Commit changes' dialog box. It contains two text input fields: 'Finish README' and 'And mention moon tacos'. Underneath these fields are two radio button options:

- Commit directly to the `readme-edits` branch
- Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

At the bottom of the dialog are two buttons: a green 'Commit changes' button and a red 'Cancel' button.



Step 4. Open a Pull Request

Pull Requests are the heart of collaboration on GitHub. When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show *diffs*, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

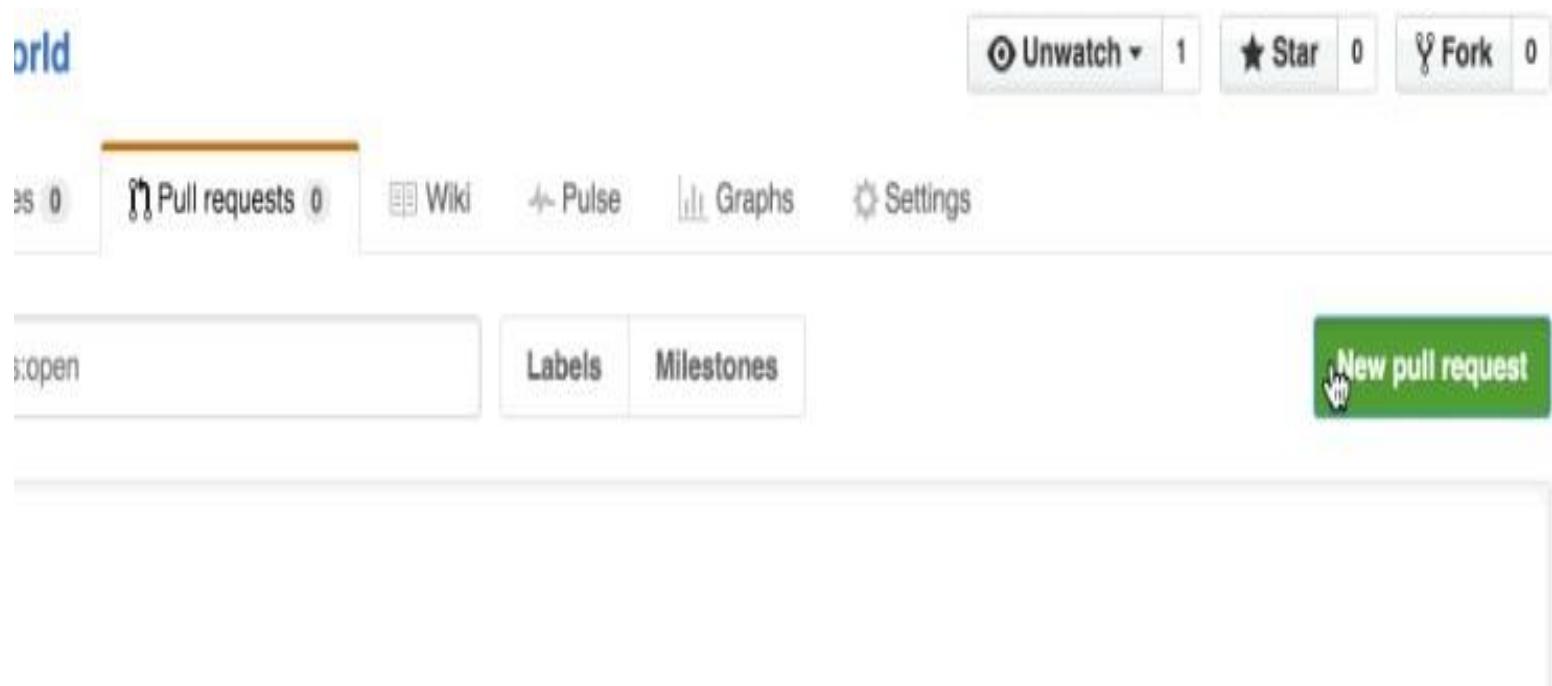
By using GitHub's [@mention system](#) in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

You can even open pull requests in your own repository and merge them yourself. It's a great way to learn the GitHub flow before working on larger projects.



Open a Pull Request for Changes to the README

Click the Pull Request tab, then from the Pull Request page, click the green New pull request button.





Open a Pull Request for Changes to the README

In the **Example Comparisons** box, select the branch you made, `readme-edits`, to compare with master (the original).

EXAMPLE COMPARISONS

 **readme-edits**

4 minutes ago



Open a Pull Request for Changes to the README

Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.

1 commit 1 file changed

Commits on Oct 27, 2014

hubot Finish README ...

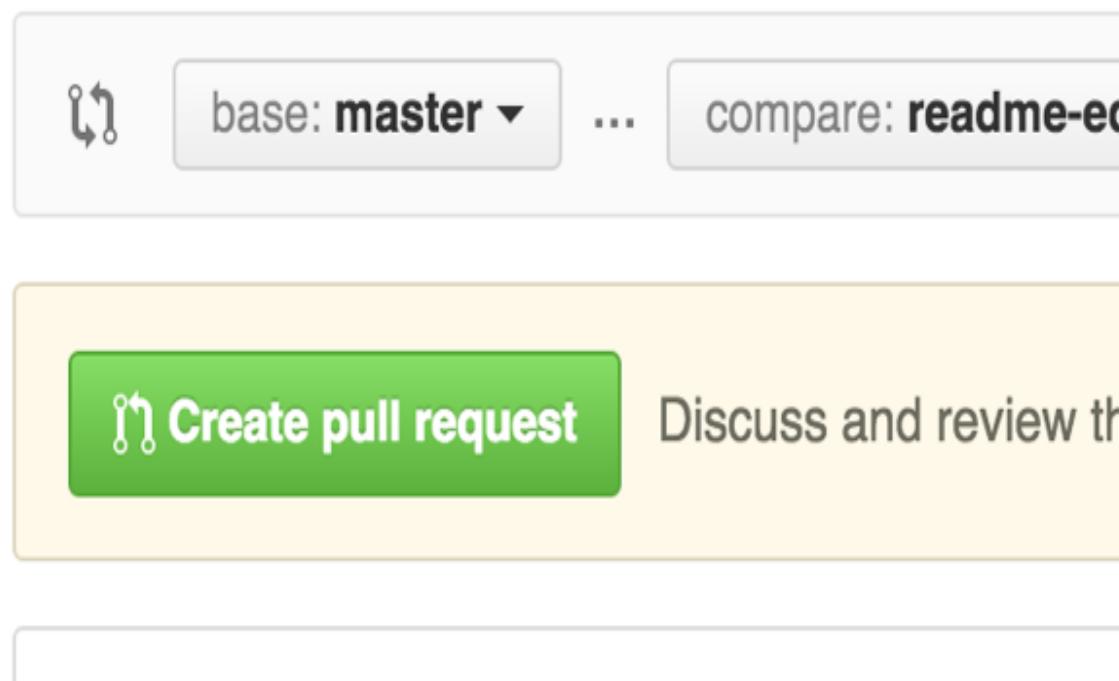
Showing 1 changed file with 1 addition and 1 deletion.

2	1	README.md
...	...	@@ -1,4 +1,4 @@
1	1	hello-world
2	2	=====
3	3	
4	4	-Just another repository
		+Hubot here, I like Node.js and Coffee them far superior to Earth tacos.



Open a Pull Request for Changes to the README

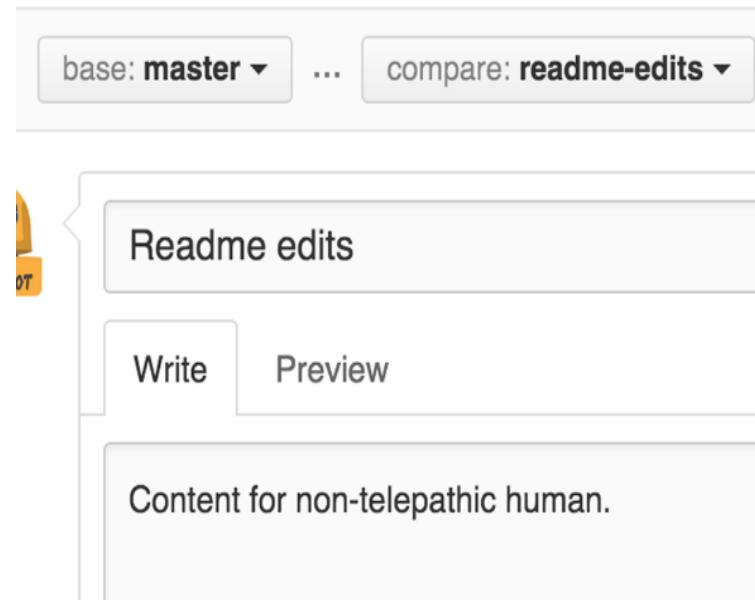
When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.





Open a Pull Request for Changes to the README

Give your pull request a title and write a brief description of your changes
When you're done with your message, click **Create pull request!**





Step 5. Merge Your Pull Request

In this final step, it's time to bring your changes together – merging your `readme-edits` branch into the `master` branch.

1. Click the green **Merge pull request** button to merge the changes into `master`.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.

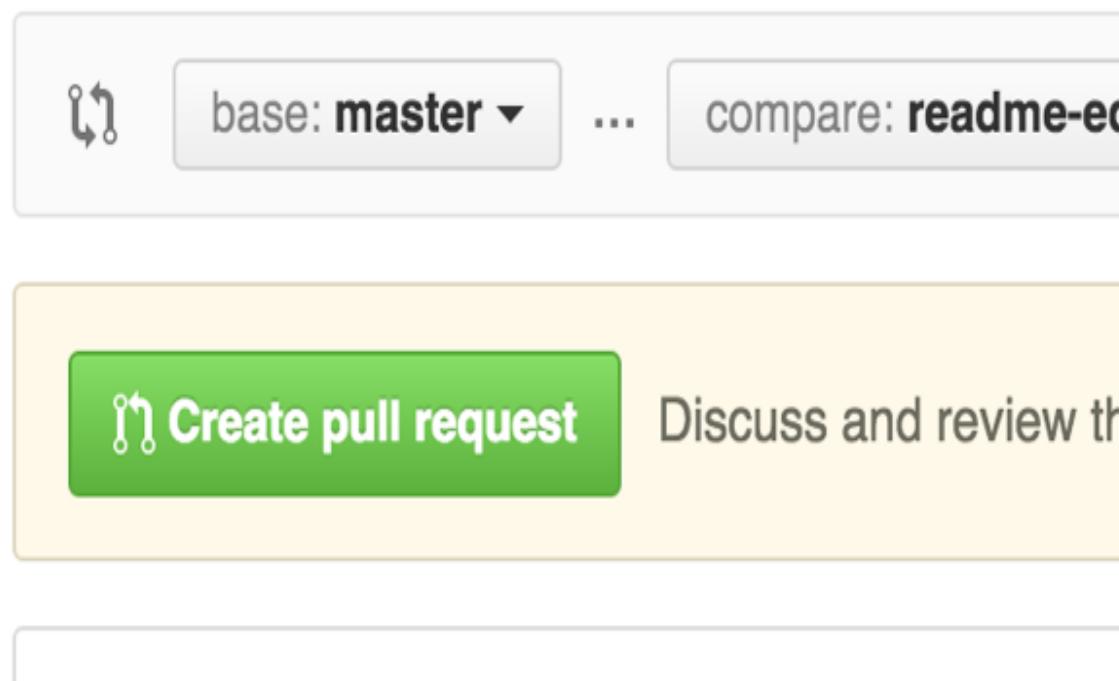
The screenshot shows two steps of the merge process:

- Step 1: Pre-merge status**
 - A green icon with a wrench and gear is shown next to a green circle with a checkmark.
 - The text "This branch has no conflicts with the base branch" is displayed.
 - "Merging can be performed automatically."
 - A green button labeled "Merge pull request" is present.
 - Below it, text says "You can also open this in GitHub Desktop or view command line instructions."
- Step 2: Post-merge status**
 - A purple icon with a gear and checkmark is shown.
 - The text "Pull request successfully merged and closed" is displayed.
 - "You're all set—the `readme-edits` branch can be safely deleted." is shown.
 - A purple button labeled "Delete branch" is present.



Open a Pull Request for Changes to the README

When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.





Done!

References:

<https://www.atlassian.com/git/tutorials>

[https://www.atlassian.com/continuous-delivery/continuous-integration#:~:text=Continuous%20integration%20\(CI\)%20is%20the,builds%20and%20tests%20then%20run.](https://www.atlassian.com/continuous-delivery/continuous-integration#:~:text=Continuous%20integration%20(CI)%20is%20the,builds%20and%20tests%20then%20run.)

<https://git-scm.com/>

<http://thehackernews.com/2015/03/github-hit-by-massive-ddos-attack.html>

FamilyMealPlanningAppX

Sue is a dietitian who has been working for 25 years. During her career, she has gained much experience and knowledge in helping individuals with both managing their weight and their different dietary requirements. With so much conflicting information out there (such as the internet and other non-qualified ‘experts’), it is easy for individuals (even those working with a dietitian) to become confused about what they should be eating, and what healthy eating is all about.

When Sue began her career, most of her clients were interested in losing weight and she assisted by both educating her clients, as well as helping them by preparing a meal plan that fit their lifestyle and dietary requirements, in order to help achieve agreed upon weight goals. As the years passed, she noticed that her clients’ needs have become more complex, especially due to an increase in individuals suffering from different food allergies and intolerances. She has also noticed that people’s eating habits have changed, and more clients are wanting to become vegetarian or vegan and others will only eat organic foods. As such, she is constantly needing to modify her information booklets and her meal plans to account for specific dietary requirements and intolerances.

Here is some information on some of the dietary requirements (including allergies and intolerances for certain food), that clients present with, in a typical work week:

Food Allergies

A food allergy is an abnormal immune response to food, which is not to be confused with food poisoning. This abnormal response ranges from mild (for instance, itchiness) to severe (known as anaphylaxis). Severe responses can be life threatening. This response can be immediate or several hours after exposure. The cause of food allergies is unknown. In some instances, allergies experienced during childhood may be resolved when the person becomes an adult. Common allergies include peanuts, eggs, shellfish, however this varies by location^[4]. In Australia, food allergies are estimated to affect 1-2% of adults and 4-8% of children under 5 years of age, with as much as 10% of children under the age of one^[5]. Another common allergy is related to gluten. Non-celiac gluten sensitivity (NCGS) (gluten sensitivity) is an immune reaction. Once foodstuff containing gluten is removed, intestinal and/or extraintestinal symptoms improve^[3]. Sue does not diagnose food allergies, but rather refers clients to a specialist if they complain of certain symptoms. It is not simple to diagnose an individual with a food allergy and it can require a medical history, elimination diet, skin prick test, blood tests for food-specific IgE antibodies, or oral food challenges^[5].

Food intolerances

Food intolerances are different to allergies, in that they are not immune responses. They refer to digestive problems that occur after certain foods are eaten. People suffering from food intolerances typically remove or restrict those items from their diet, in order to avoid the consequences of eating those foods. A low FODMAP diet consists of restricting or removing fermentable carbohydrates (FODMAPS) from one’s diet. This sort of diet is recommended for people suffering from irritable bowel syndrome (IBS)^[1]. Similarly, lactose intolerance is quite a common intolerance caused by a decreased ability to digest lactose; a sugar found in dairy products. Individuals who suffer from lactose intolerance, have varying amounts of lactose they can tolerate, before symptoms develop^[2].

Vegan and vegetarian

For the purposes of understanding veganism as it relates to Sue as a dietitian, we will limit this explanation to veganism with respect to dietary choices only, with a focus on some of the different variations (and no discussion on the reasoning behind the variations). Vegans do not eat meat, eggs, dairy products and any other animal-derived substances. Strictly this also includes honey. Some people will eat eggs, milk and honey^[6]. A pescatarian is

someone who will eat fish and seafood in an otherwise vegan diet^[8]. Veganism is not generally recommended for children or adolescents, for women during pregnancy or for a woman breastfeeding. Some people look for substitutes such as tofu^[6], and some turn to a dietician to help ensure that their bodies are getting all the nutrients and protein that their bodies need.

Diabetes

Sue usually tries to remove or limit sugar from meals to help clients lose weight and eat in a healthier way. She also has some clients that suffer from diabetes, which is a group of metabolic disorders characterised by high blood sugar^[7]. There are different types of diabetes and sufferers will see an endocrinologist^[6] who will provide information as well as dietary modifications. Sue has some clients who suffer from diabetes, and clients who have family members who suffer from diabetes, and are concerned that they may be predisposed to the disease.

Organic food

Organic food is food grown and produced through following standards set out for organic farming. The production may include practices on recycling resources and/or only using specific pesticides and fertilisers^[9].

As these dietary requirements may only influence one member of the family, the family has a few options.

1. Everyone eats the same meals based on the meal plan created with the help of the dietician.
2. The family eats the same as before and a separate meal is made for the individual with the meal plan.
3. The family eats the same as before and substitutions are made for the member of the family who cannot eat an item/s in the recipe.

Here are a couple scenarios explaining the different options that a family could have when one member of the family follows a specific meal plan,

Scenario 1- The family is having spaghetti bolognese for dinner and one member of the family does not eat onions or tomato as part of a low FODMAP diet.

The person cooking the meal may remove some meat and make that separately with other flavours and spices (option 3). Another option would be that everyone in the family eats the spaghetti bolognese without onions and tomato (option 1). The family member with the meal plan can eat something completely different, as that family member may think that spaghetti bolognese isn't really spaghetti bolognese without onions or tomato (option 2),

Scenario 2- The family is having spaghetti bolognese for dinner and one member of the family is vegan.

All members of the family would eat a vegan meal plan if one member of the family becomes vegan, so in this instance they could use a meat-free substitute and make the spaghetti bolognese as before (option 1). Another option would be for the person cooking the meal to make a portion with the meat-free substitute and the rest as before (option 3). There is also option 2, where the family member who is vegan can eat something different.

Sue wants to create a family meal planner that will help clients such as these described in the scenarios above. She also believes there is a market for family meal planners, based on her experiences with the clients she has helped over her 25 years in the industry.

She knows that some people have chosen to use services such as 'Lite n Easy' or 'Marley Spoon', but she has many clients who like to cook and want to be involved in meal preparation. She believes individuals will pay for the convenience because, they appreciate that the complexity of family meal planning has been removed. Sue sees an opportunity to create an app that will help individuals manage their diets and their family meals effectively

based on their food preferences and food intolerances/allergies. Although there are many websites and apps that address these issues, she has not yet found one that provides a holistic response and that she can direct clients to. She wants to create an app that will help simplify meal planning. She intends to give the app to her clients for free, but at a cost to anyone else who wants to download it.

Hue (Sue's daughter) is an entrepreneur and has brainstormed ways to market the app and make it (more) profitable. She sees a market for some niche companies to advertise. For instance, companies that create food substitutes, such as brands of tofu or different types of non-dairy milk could be interested in advertising. Gluten free stores and restaurants (e.g., vegan restaurants) may also want to advertise as this may be a good way to increase their client base.

Sue and her daughter Hue have discussed and decided to each contribute \$15000 (total= \$30000) on an initial project to create an app for Sue's business. In the first phase, after the development, they plan on releasing the app and will wait 6 months after the release, to see if their prediction is correct and there is a gap in the market – that there is a demand for this FamilyMealPlanning app. If they decide that it is a worthwhile venture, they will contribute more money for a second development where they will add more functionality. They have approached DevCo to do the development. DevCo have decided to take on the initial project and have agreed to complete the initial project in 6 months. The first stage of the project is the creation of 'FamilyMealPlanning' app. These are the key elements of this initial development phase (phase 1).

The user should be able to create a Family Profile with basic signup information including-

- email, password,
- and a name (the name they wish to be referred to).

The profile should be a family profile and should contain each member of the family (for that single logon). For each family member there should be an option to add dietary requirements. The system should provide dietary options (e.g. lactose intolerant, gluten free, sugar free etc) that the user can choose. These options should be clear and the process of adding the data should also be intuitive. For instance, an option could be 'No sugar'. In this instance it should be clear if the requirement is for diabetes or healthy eating/weight loss. Linked to this the user should be able to add a substitute(s) if necessary. For instance, if one of the family members is lactose intolerant, the user can add 'almond milk' as a substitute.

All this data is manually added by the user when they sign up to the app. The user can modify the data at any time. For instance, a family member may initially be a vegan who eats eggs, but at a later date change their mind and no longer eat eggs. Similarly, a child may have an allergy to nuts which they outgrow. The interface should allow for modifications.

The user should be able to upload recipes to the app to use in their FamilyMealPlanning. The format of the upload can be .pdf, .doc, .txt. The user should also be able to add a URL link with the recipe. These recipes should all be stored and should have the functionality to be added to a calendar. This calendar will be the basis of the FamilyMealPlanning app. When the user opens up the calendar, they can view what recipe they are using to cook dinner that night (from the list of recipes uploaded).

In this initial development (phase 1), the user should be able to click on the recipe in the calendar. When the user views and uses the recipe, it should provide suggestions for substitutions to the ingredients (for instance replace milk with almond milk if one of the family members is lactose intolerant), based on the dietary options set up during the family profile creation. Please remember: this is an initial development of the product (phase 1) and **does not include all the functionality described in the case scenarios.**

About the company DevCo: DevCo was formed by 6 students who completed their Master of Information Technology degree at the University of Melbourne in 2019. Due to COVID-19, many of the projects they spent time on were helping clients set up their remote working environments, and providing tech support. They are also involved in and completing a few other development and upgrade projects for some of their clients. They are excited and eager to start this project.

Please answer the following questions based on the information in the case described above.

Section 1: Short Research Questions

1. Identify a major goal for the *initial development* of the software product. (maximum 50 words – 1 mark)
2. Identify the value of the software product for Sue. (maximum 50 words – 1 mark)

Section 2: Extended Research Questions

3. Identify 3 **high** level features (functions the IT solution should provide) in the initial software product (*the initial development ONLY*) and categorize their priority as high (must have), medium (good to have) or low (can do without). Give a justification for the chosen priority of each requirement. (3 marks) You can use the following table with an example of a high-level feature (which does not count as one of your 3 features).

Requirement	Priority	Justification
A user can set-up a family profile	High	The user needs to set-up a family profile to use the functionality provided by the FamilyMealPlanning app. This is a core feature of the app and high priority as without it, users cannot set-up their dietary requirements and the app will not be able to provide any recommendations or modify recipes based on dietary requirements.

4. Identify two challenging characteristics (characteristics that would make the project difficult) of this IT project (*the initial development*) and discuss why you consider them challenging. (maximum 200 words – 2 marks) *
5. Identify and justify three things that can go wrong – *risks*, resulting in the project (*the initial development*) not achieving the intended goal/s. Ensure that you identify risks that are unique to the characteristics of this case study, rather than generic risks that can occur in **any** project. Examples of generic risks are project members leaving the project; running out of budget before completion; IT security risk, a good user

interface (please note: this is not an exhaustive list of generic risks). Describe the *impact* each of these risks could have on the business, project and/or the software product. (maximum 250 words - 3 marks) *

*For both Q4 and Q5, please ensure that by your explanation of the characteristic it is clear that it is a challenge and not a risk (Q4) and a risk and not a challenge (Q5).

Section 3: Discussion

6. Discuss two possible lifecycle models you would consider for the project SDLC model. This should include the pros and cons of each of your choices referring to specific project characteristics and risks you have identified. Use case study references to support your argument. Please do not provide a lengthy description of the models, you can assume the marker has basic understanding of the model. Furthermore, use outside references (at least 2 references) to further expand your discussion (maximum 800 words – 6)
7. From the 2 possible choices you described in Q6, choose the most suitable SDLC and justify your choice referring to specific project characteristics and risks you have identified. (maximum 200 words – 2 marks)

Section 4: Ethics

8. Question to be released at a later date and due week 12 (2 marks)

Quality and presentation: -2 marks

Please note: I have used Wikipedia references to explain terms that may not be familiar to students. I caution students to think before using Wikipedia in your responses. While the references used in the case study are to illustrate some non-familiar and domain specific concepts in a lucid and simple manner, academic arguments (such as your answers in the Assignment) should make use of scientific references only. For more information on referencing, please refer to the university guidelines (and additional resources) at the link below.

[recite \(unimelb.edu.au\)](http://recite.unimelb.edu.au)

References

[1] (13 January 2021). FODMAP [Online].

Available: <https://en.wikipedia.org/wiki/FODMAP>

[2] (13 January 2021). Lactose Intolerance [Online].

Available: https://en.wikipedia.org/wiki/Lactose_intolerance

[3] (13 January 2021). Non-celiac Gluten sensitivity. [Online].

Available: https://en.wikipedia.org/wiki/Non-celiac_gluten_sensitivity

[4] (14 January 2021). Food Allergy. [Online].

Available: https://en.wikipedia.org/wiki/Food_allergy

[5] (14 January 2021). Allergy and Anaphylaxis Australia. [Online].

<https://allergyfacts.org.au/allergy-anaphylaxis/food-allergy>

[6] (25 January 2021). Veganism. [Online].

Available: <https://en.wikipedia.org/wiki/Veganism>

[7] (25 January 2021). Diabetes. [Online].

Available: <https://en.wikipedia.org/wiki/Diabetes>

[8] (25 January 2021). Pescetarianism. [Online].

Available: <https://en.wikipedia.org/wiki/Pescetarianism>

[9] (25 January 2021). Organic Food. [Online].

Available: https://en.wikipedia.org/wiki/Organic_food

[10] (25 January 2021). Pesto. [Online].

Available: <https://en.wikipedia.org/wiki/Pesto>



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

2021 – Semester 1
Lecture 2

Copyright University of Melbourne 2020

What is the NHS? (Understand the context)

UK National Health Service

(publicly-funded healthcare systems)

What was the project?

How a large teaching hospital developed, executed and practically implemented a KMS (knowledge management system).

Electronic Patient Record (EPR) system – medical records stored in digital form

[1]

Case Study

WILLING TO LEARN

What was the motivation for the project

Knowledge available to clinicians at the point of care to enhance the communication and decision making processes.

(A move to a new facility and the corresponding lack of space for paper records presented a problem)

IT

Commercial product that was customised by the Trust's own ICT staff

[1]



WILLING TO LEARN

Development and implementation was overseen by **EPR Board**, which comprised a mix of senior managers, IT professionals and clinical leads

Initial pilot phase (feasibility)

There was no formal written communication plan

Research conducted

A qualitative study was conducted over a 2.5 year period with data collected from semi-structured interviews with
8 members of the strategic management team,
12 clinical users and
20 patients
(in addition to non-participant observation of meetings and documents)

[1]



Findings

- Identification of business problem to be solved and alignment of the KM project with overall business objectives. (Business case)
- Communication strategies should heavily focus on transparency including both structured and unstructured
- Communication methods.
- Communication strategies play a large role in changing attitude and mentality of workforce.
- Opportunities to highlight and sell positive messages about the KM initiative
- Engage all workers and address issues and concerns (users)

[1]

[1] Sara S. McCracken and John S. Edwards, “Implementing a knowledge management system within an NHS hospital: a case study exploring the roll-out of an electronic patient record (EPR),” *Knowledge Management Research & Practice*, vol. 15, pp. 1-11, 2015.



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

2021 – Semester 1
Lecture 1

Copyright University of Melbourne 2021



Your Subject Support Staff

Lecturer: [Marion Zalk](#)
(Subject Coordinator)



Tutors



*Esther
Rotimi*



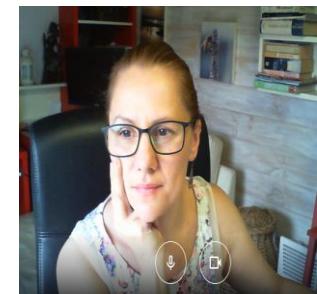
*Rajesh
Sundaram*



*Doc
Wallace*



*Eileen
O'Callaghan*



*Renata
Romila*

Marion Zalk – Subject Coordinator

Academic – Melbourne lecturer



Professional Experience

- Senior Programmer/Analyst
 - Financial and wealth management organizations (Citigroup in London, UBS & ComputerShare in Melbourne, MCubed); Consultant
- 8 Years experience
 - Waterfall, agile, hybrid
- PhD (University of Melbourne)



Where's Everyone?





Poll - Tutors and You



AIM

The aim of this subject is to introduce students to the software engineering principles, processes, tools and techniques for analysing and managing complex software projects.

INTENDED LEARNING OUTCOMES

- Select appropriate software engineering processes and practices projects
- Manage team dynamics and professional communication
- Plan and manage projects
- Identify risks and modify project activities to mitigate these risks
- Manage software project activities to ensure a quality product
- Describe human and organisational implications of change and explain the organisational change process.

GENERIC SKILLS

- In-depth knowledge project management areas of knowledge
- Reach a high level of achievement in writing, research or project activities problem-solving and communication through the writing of project analyses
- Ability to function effectively as an individual and in a multidisciplinary and multi-cultural team as a leader, manager or effective team-member
- Understand and respect ethics and intellectual integrity
- Writing and communication skills.



LECTURES

- attend
- available via LMS

**Tutorials Start
Friday
Week 1**

TUTORIALS / WORKSHOPS

- Students are expected to attend one one-hour workshop/tutorial each week.
- Workshops are intended to take concepts and principles discussed in lectures and to apply them to realistic examples.
- Students are expected to actively engage in workshops and show initiative, ask questions, conduct workshop exercises and engage in the discussions

PLAGIARISM AND COLLUSION

- Submissions must be own independent work or group project work.
- Faculty policy: mechanisms for establishing contributions of individuals to group work
- University policy: see <http://academichonesty.unimelb.edu.au/policy.html>
- If plagiarism or collusion is detected, harsh penalties must and will be applied

Communication

How to Contact Us

- General inquiries: Discussion forum on LMS
- We encourage all students to join in discussions – answering other students' questions is one of the best ways to improve your own understanding
- Please do not post sections of your assignments- if you must include these, private-message the instructors
- Personal/private concerns: Email the lecturer
- If you email us about a general inquiry, we may ask you to re-post your question in the forum
- Please include SWEN90016 in email subject



Due dates are available in LMS- Assignments, Quizzes (3 for marks, week 5, 9, 11) and other quizzes for review.

I'll be discussing Assignments in more detail next

Zoom

Please update your computer to the latest version to make use of the features

Cameras- I am recording each live lecture for lecture capture



Poll- Face to face workshop

Feedback has been mostly positive, a few suggestions for improvement.

1. General

- Would like to thank all teaching staff in delivery of quality content in a tough period! The presenters both lecturing and tutoring were great.
- Make it an elective, not core.
- Having mid-term break on week 9 is really a bad idea,
- I found my two units (SWEN90016 and xx) to be engaging and impactful - especially being my first semester in post-graduate study.
- The harder, more complex topics were given equal time as the more simple topics, an extra week or workshop reviewing them (project scheduling and cost estimation)
- High interaction based learning with integrated quizzes within the lectures and group discussion based learning in tutorials
- The material was laid out very well, and all of the teaching staff did their best to engage all of the students and make sure nobody was left behind conceptually
- X's mic makes it slightly hard to follow.
- Person's accent difficult to follow for international students.
- Practical learning

Feedback has been mostly positive, a few suggestions for improvement.

2. Lectures

- For online learning I also prefer shorter 1 hour lectures. 2 hour zoom lectures are too long, especially in the evening it is very easy to lose focus.
- The lectures are not engaging and interactive. Too much talking on concepts instead of how to apply them in real life.
- The lecturers and tutors are working hard for presenting a subject with good quality.
- The lecture slides are clear on knowledge of the topic.
- The online learning experience was just fabulous. Particularly the lecture recordings and the study materials provided by the teaching staff was highly effective and did me to a great extend.
- Staff represent project management in the slowest changing sectors, government, health and banking.
- I can see that the materials is constantly being updated and also the teaching team try to keep it relevant to the current working conditions.
- Knowledge in lecture

3. Exam revision

Feedback has been mostly positive, a few suggestions for improvement.

4. Assignments

- Assignments were interesting
- Create open sessions for us to discuss assignment and ask broad questions
- Clear assignment requirements
- The assignment's requirements should be clearer. And it would be really helpful for us to get some examples of the assignments. Since the students have various academic background, some may be familiar with what the assignment is talking about, others probably even have no idea to do a draft. Although we did researches, investigation and asked in discussion board, an example would be more useful. The example can give us a direction to do a correct thing.
- Learned more about how to plan and manage the development process.
- Teamwork in writing group project.

Feedback has been mostly positive, a few suggestions for improvement.

5. Tutor and tutorials

- Tutor X doesn't seem to enjoy teaching and rush through everything
- I felt the tutorials were run fairly interactively and there was a reasonable amount of practice and hands on learning that worked well with the subject material.

6. Discussion boards

- Thank you Marion for putting efforts into building such an efficient and helpful discussion board for students
- The subject coordinator would reply very soon and willing to answer our questions.

7. Quizzes

- I liked the quizzes during the semester, as they helped me to stay up to date with the content and make the current exam preparation much easier.
- Intermittent tests were a great way of keeping up to date with the content. I wish more subjects had this



Weeks 5, 9, 11

- 15 minutes in a 30 minute window;
- Discuss the solutions
- During the lecture time (open 15 minutes before and then first 15 minutes of the lecture)
- Lecture will start 6:40pm (week 5, 9, 11)



Assignment 1 - Learning Outcomes:

Analysis of an IT project case study that will demonstrate the ability to:

- identify the goals of the project;
- identify the key characteristics for the project;
- identify the risks in the project as identified at the start of the project; and
- justify the choice of a suitable software development lifecycle (SDLC) model to manage the project.



Assignment 1:

- Assignment specification will be given out by the end of week 2 – due week 5
- You are expected to spend ~30 hours
- You will answer a set of questions related to a given case study
- Will be based on the material covered in weeks 1-4



Assignment 2 - Learning Outcomes:

A web-based IT system development project that will demonstrate the ability to:

- develop a Project Management Plan (PMP) for a given project brief;
- plan the activities involved in the chosen process;
- execute, monitor and control processes to achieve an outcome; and
- work effectively in a team.



Assignment 2

- Teams:
 - We will organise teams of 4-5 members during week 2
 - Week 2 workshop attendance is mandatory as we will be selecting teams
 - Each student is expected to spend 30-40 hours on the project
 - If you have not done the foundation subjects in your degree (or have received credit for them) please consider doing this subject next semester, after you complete the foundation subjects

Assignment 2:

- Assignment specification will be given out by the end of week 2 – due week 7, 11 and a presentation in week 12
- You are expected to spend ~30 hours
- Will be based on all the material



Class rep

If you are interested in becoming class representative

Please email me

mzalk@unimelb.edu.au



Poll- Assignments

Lecture 1 – Intended Learning Objectives

Module 1: Subject Introduction

Module 2: Introduction to Projects

Module 3: Projects



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2017

2021 – Semester 1
Lecture 1

Lecture 1 – Intended Learning Objectives

Module 1: Subject Introduction

1. ~~Get to know your teaching staff, subject learning objectives, subject content and semester structure.~~

Module 2: Introduction to Projects

1. Understand key elements of a Project and why organisations use them.
2. Understand the foundational components of Project Management.
3. Understand key skills and responsibilities / activities of a Project Manager.

Module 2.1 – What is a Project

A temporary endeavour to create a unique product, service or outcome.

Key characteristics:

- Introduce **CHANGE** to the organisation
- **TEMPORARY**, it has a defined beginning and end
- **CROSS-FUNCTIONAL**, cuts across organisational boundaries
- Deals with the **UNKNOWN**
- **UNIQUE**
- They all vary in **SIZE** –  /  , \$'s and 

Module 2.1 – Why do organisations use Projects

- Provides strategic alignment of key activities and visibility at the appropriate levels
- Mechanism to prioritise activities (Benefits, Regulatory, HW Refresh)
- Allows organisations to deliver change in a structured and formal manner outside of BAU
- Effective and efficient management of organisations limited resources (people & \$'s)
- Establish ownership and accountability – Process and the Benefits
- Provide clarity, buy-in and agreement across what will be done, when, who, why and the outcomes

www.pmi.org/about/learn-about-pmi/what-is-project-management



Lecture 1 – Intended Learning Objectives

Module 1: Subject Introduction

1. Get to know your teaching staff, subject learning objectives, subject content and semester structure.

Module 2: Introduction to Projects

1. Understand key elements of a Project and why organisations use them.
2. Understand the foundational components of Project Management.
3. Understand key skills and responsibilities / activities of a Project Manager.



Module 2.2 – What is Project Management

Project Management is the planning, delegating, monitoring and controlling of all aspects of a project, and motivating those involved to achieve the project objectives within the expected targets for time, costs, quality, scope, benefits and risks.

Value lies in:

- Organising and structuring scarce resources
- Managing risk
- Identifying and clearing issues
- Managing and implementing change
- Retaining and re-using knowledge
- Organisational wide learning from past success and failures



Lecture 1 – Intended Learning Objectives

Module 1: Subject Introduction

1. Get to know your teaching staff, subject learning objectives, subject content and semester structure.

Module 2: Introduction to Projects

1. Understand key elements of a Project and why organisations use them.
2. Understand the foundational components of Project Management.
3. Understand key skills and responsibilities / activities of a Project Manager.

Module 2.3 – Project Manager Skills / Attributes

Project managers are highly skilled knowledge workers and change agents. They take accountability, make project goals their own and use their skills and expertise to inspire a sense of shared purpose across the project team. They enjoy the organised adrenaline of new challenges and the responsibility of driving business results.

Core Skills / Attributes:

- Work well under pressure
- Comfortable with change and complexity in changing environments
- Use / have the right people skills
- Adapt, resolve issues and deal with problems
- Effective communicators regardless of hierarchy
- Action orientated and leave nothing for tomorrow
- Command & Control
- ***Good ones are in demand, hard to find and get paid a lot***

www.pmi.org/about/learn-about-pmi/who-are-project-managers

Module 2.3 – Project Manager Key Activities (traditional)

Planning

- Define and clarify project scope
- Develop the project management plan
- Develop the project schedule
- Develop policies and procedures to support the achievement of the project objectives

Leading

- Setting team direction
- Owning & coordinating activities across different organisational functions
- Motivating team members
- Assigning work

Organising

- Determine the project team structure
- Identify roles and responsibilities
- Identify services to be provided by external companies
- Staff all project positions and on-going management

Controlling

- Defining project baselines
- Tracking project progress
- Project status reporting
- Determining and taking corrective actions

Module 2.3 – Agile Scrum Master Key Activities *“a change is occurring”*

Agile is redefining the way we execute projects and the role of the PM.

In pure Agile:

- No defined PM role
- Key activities are spread / shared across team members
 - Key project activities are still undertaken formally with appropriate documentation
- Some alignment between a Scrum Master and a Project Manager
- Move from Command and Control to Servant Leadership
 - Coaches and facilitates teams to deliver
 - Emphasises objectives
 - Is invested in the program's overall performance
 - Asks the teams for answers
 - Allows the teams to self-organise and hit their stride
 - Assists others with fixing issues

www.pmi.org/learning/library/pm-role-lean-agile-world-9350

www.greenleaf.org/what-is-servant-leadership/

<https://www.mountaingoatsoftware.com/agile/agile-project-management>

Module 2.3 – Project Manager Key Activities – *The Market wants it all!*

Project Manager Job Ad - Skills and experience

- Minimum 8+ years of experience as a Project Manager managing large, complex projects with multi-functional teams
- Strong stakeholder and relationship management skills
- Experience in managing multi-vendor teams
- Experience and qualifications in Prince2
- Can deal with complexity with very solid Project Management technical skills such as ability to develop and manage schedules, financial workbooks
- Strong stakeholder partnership skills, ability to work with teams at varying levels of project experience
- Key requirement is knowledge of multiple models of technical project delivery such as agile and running sprints but equally able to build confidence with the steering committee with formal project management approaches such as setting and achieving deadlines on timing and scope



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2017

2021 – Semester 1
Lecture 1

Lecture 1 – Intended Learning Objectives

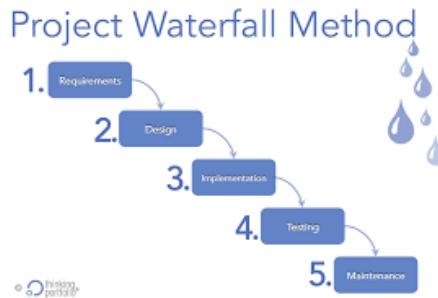
Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.

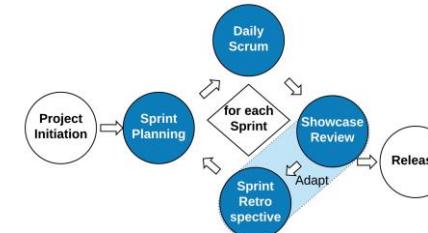


Module 3.1 – PM Methodologies / Standards

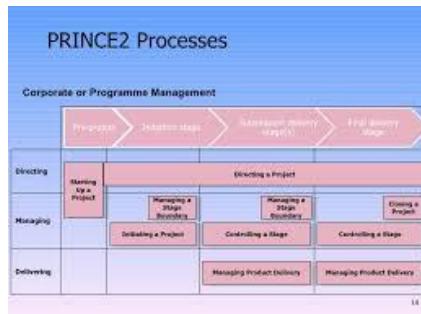
Waterfall



SCRUM



Prince2



Agile



PROJECTMANAGER.com



Module 3.1 – Key Elements of Project Management Methodologies / Standards

Waterfall

- Traditional approach used for over 40 years
- Requirements must be defined at the start
- Little / no alternations
- Sequential - Complete 1 task and then the next
- Used in large scale SW development where thorough planning and predictability is required

Pros

- Extensive planning, this thoroughness often results in more accurate timelines and budgets

Cons

- Difficult to apply changes or modify / correct previous steps (water can't run backwards), need to be proactive in anticipating problems

Module 3.1 – Key Elements of Project Management Methodologies / Standards

Agile

- Focuses on adapting to changing situations
- Reliant on constant and regular feedback
- Focuses on iterative outcomes delivering value as quickly as possible & collaboratively
- Small manageable actions and activities
- Involvement & ownership across the team – Team members self select work
- Customer focus over formalised sign-offs

Pros

- Retains flexibility while continually producing outcomes – less rework
- Greater communication & engagement – increased buy in across the team of the end outcome

Cons

- Difficult to do without experience – especially an experienced Scrum Master
- Large projects co-location a problem
- Difficult to contract suppliers

Module 3.1 – Key Elements of Project Management Methodologies / Standards

Structured Project Management Methodologies e.g. PRICNE 2 etc

- Widely used and accepted - Consulting, Private and Government
- Process orientated approach
- Divides projects into multiple stages
- Detailed and thorough
- Must have a clear need, a target customer, realistic benefits, and a thorough cost analysis

Pros

- Extensive documentation is helpful with corporate planning & tracking

Cons

- Difficult and untimely to adapt changes and apply these to all documentation



Module 3.1 – Project Methodologies – Which one is the right one?

- They all have a place and all can be appropriate
- It is like selecting the best recipe – *it all depends on your ingredients*
- Items (ingredients) to consider include:
 - Clarity and stability of scope
 - Timelines
 - Tools to support / drive the process
 - People / knowledge
 - Organisational maturity
 - Stakeholder buy-in
 - Experience in the various approaches

BREAK

Please return promptly as the

Lecture will re-start in *10 mins*



Original estimate

- \$1.2m
- 12 months

Final outcomes

- \$2m (60% increase)
- 18 months (50% longer)

FAILURE ----- SUCCESS



Redefined the market in Tracking, Pricing, Staff Pay, Customer Flexibility and Transparency

Original estimate

- \$1.2m
- 12 months



Final outcomes

- \$2m (60% increase)
- 18 months (50% longer)

‡ Recent News & Activity

↗ Acquisition • Nov 19, 2015

Royal Mail acquired eCourier.co.uk for an undisclosed amount

FAILURE ----- SUCCESS

BREAK

Please return promptly as the

Lecture will re-start in *10 mins*

Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.

Module 3.2 – Project Success / Failure – You decide

Failure or Success?

- Original estimate
 - \$1.2m
 - 12 months
- Final outcomes
 - \$2m (60% increase)
 - 18 months (50% longer)

FAILURE-----SUCCESS



Module 3.2 – Project Success / Failure – You decide

Failure or Success?

- Original estimate
 - \$1.2m
 - 12 months
- Final outcomes
 - \$2m (60% increase)
 - 18 months (50% longer)



FAILURE ----- SUCCESS



Module 3.3 – Project Success / Failure – You decide

Failure or Success?

- Original estimate
 - \$7m
 - 6 years
- Final outcomes
 - \$102m (1,357% more)
 - 16 years (10 years longer)

FAILURE-----SUCCESS



Module 3.3 – Project Success / Failure – You decide

Failure or Success?

- Original estimate
 - \$7m
 - 6 years
- Final outcomes
 - \$102m (1,357% more)
 - 16 years (10 years longer)



Module 3.2 – Software Projects

History tells us we have failed.

ALL IT PROJECTS					
	2011	2012	2013	2014	2015
Successful	29%	27%	31%	28%	29%
Challenged	49%	56%	50%	55%	52%
Failed	22%	17%	19%	17%	19%

- **Successful:** project is completed on-time and on-budget, with all features and functions as initially specified.
- **Challenged:** completed and operational but over-budget, over the time estimate or offers fewer features and functions than planned.
- **Failed:** project is cancelled at some point during the development cycle.

Standish Group Chaos Reports: Source: Standish Group 2015 Chaos Report www.projectsmart.co.uk/white-papers/chaos-report.pdf

Module 3.2 – Software Projects - What determines success?

Success Factors	%
1. Executive Sponsorship	15%
2. Emotional Maturity	15%
3. User Involvement	15%
4. Optimisation – Statement of Requirements	15%
5. Skilled Resources	10%
6. Standard Architecture	8%
7. Agile Process	7%
8. Modest Execution	6%
9. Project Management Expertise	5%
10. Clear Business Objectives	4%

- Factors have remained relatively constant
- If we know the reasons why can't we fix / improve it?
- 60% (first 4) are non technical items and difficult to change
- Broader organisational context and system at play

Standish Group Chaos Reports: www.projectsmart.co.uk/white-papers/chaos-report.pdf
www.infoq.com/articles/standish-chaos-2015

Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.

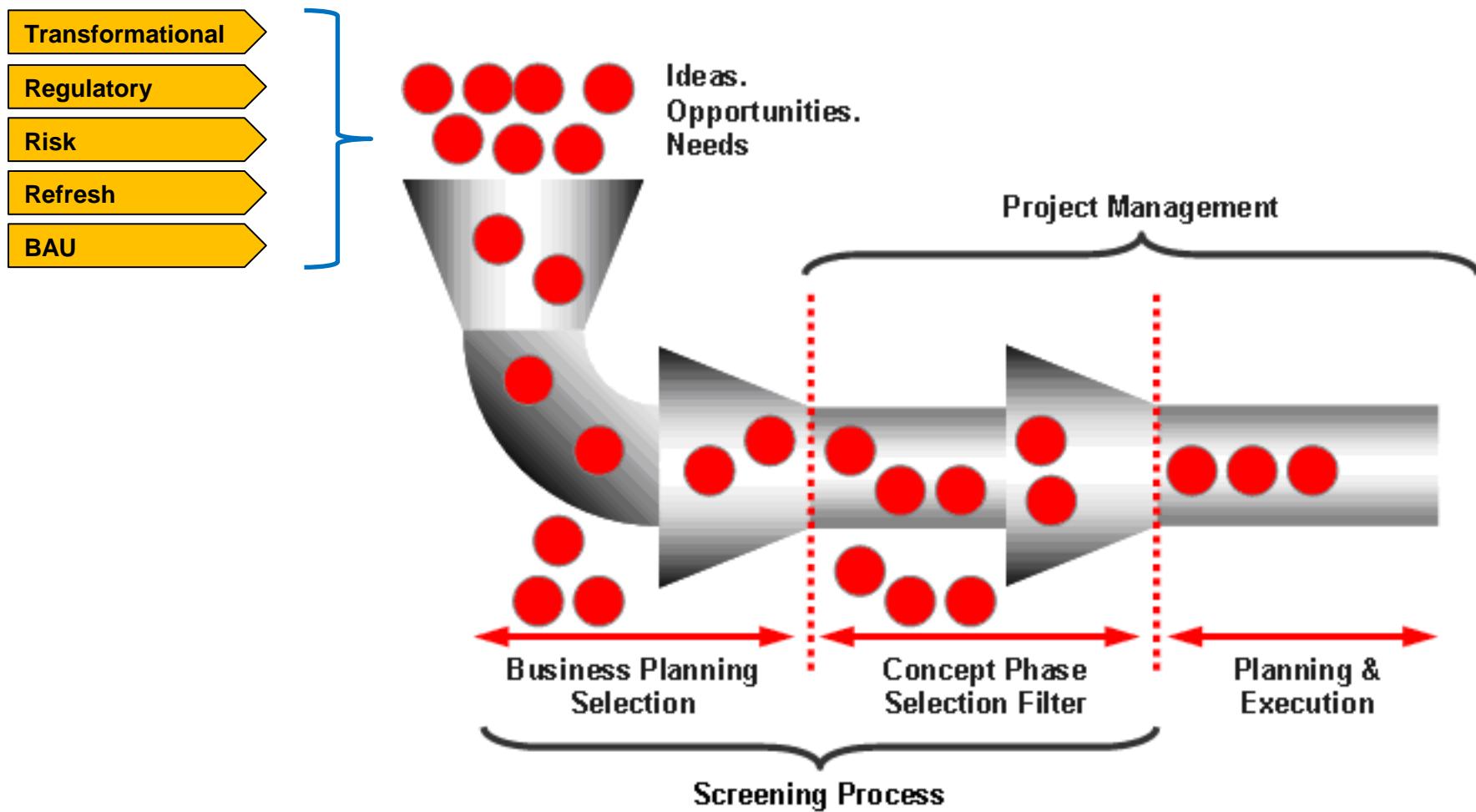
Module 3.3 – Project Screening and where to start

“If you don’t know where you’re going any road will take you there”. Any Road by George Harrison – The Beatles

- The place to start is at the beginning!
- Organisations need a formal, structured approach to:
 - Select;
 - Prioritise;
 - Have oversight; and
 - Drive accountability across all projects.



Module 3.3 – Project Screening and where to start



Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.



Module 3.4 – Project Initialization

There are many approaches and methodologies that are widely used across industry with organisations favoring standard industry ones (PRINCE2, PMBOK, Agile etc) or usually a modified version of these they make their own.

They all have Pro's & Con's.

Software PM Activities

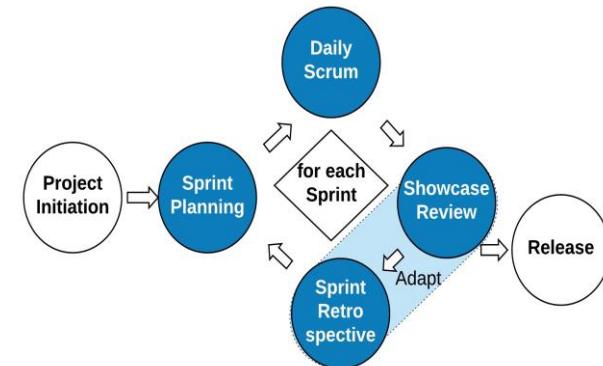


<http://blog.zilicus.com/software-project-management-activities-roles/>

Prince2



AGILE



Module 3.4 – Setting up a project for success. A Business Case is the key.

The purpose of the Business Case is to establish mechanisms to judge whether the project is (and remains) desirable, viable and achievable as a means to support decision making in its initial and continued investment.

- Provides a factual base for key decisions makers to decide if the project should be undertaken
- Demonstrates how the project adds value to the organisation
- Has a set of pre-defined standard organisational characteristics (costs, benefits, risk, etc.)
- It is not all about size - size depends on the cost / benefit
- It is a living document throughout the project that should be reviewed and signed off at key stages

Module 3.4 – Setting up a project for success. The Business case is key

Business case contains:

- Executive summary
- Reasons / explanation of why it is required
- Business options
- Expected benefits
- Expected dis-benefits
- Timescale
- Costs
- Investment appraisal
- Major risks

Source: www.prince2.com

Module 3.4 - Business Case. Who's is responsible for what?

Role	Responsibilities
Corporate	<ul style="list-style-type: none"> 1. Provides Mandate / The go ahead. 2. Holds Senior Users accountable for benefits realisation. 3. Responsible for conducting post projects benefits validation.
Executive / Sponsor	<ul style="list-style-type: none"> 1. Owns the Business Case. 2. Responsible for reviewing the benefits throughout the project.
Senior Users	<ul style="list-style-type: none"> 1. Responsible for accepting the benefits and delivering them. 2. Responsible for ensuring the delivered products are to the appropriate quality standard. 3. Provides on-going actual V forecasted benefit realisation.
Project Manager	<ul style="list-style-type: none"> 1. Prepares the Business Case. 2. Conducts Risk assessment and impact analysis. 3. Assess and updates the Business Case at each defined stage.
Project Assurance / QA	<ul style="list-style-type: none"> 1. Assists in developing the Business Case. 2. Ensure value for money and risks are continuously managed. 3. Monitors change to the Business Case and validates it.
Project Support	<ul style="list-style-type: none"> 1. Responsible for capturing data and preparing management reports. 2. Key support point for all project stakeholders – schedules, cost analysis, minutes, actions, supplier liaison etc.

Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.

Module 3.5 – It is all about the money!

- For non mandatory projects, the primary benefit is financial
- Multiple investment techniques are used to analyse the investment required / financial benefit
- Some (there are many more) techniques include:
 - Return On Investment
 - Net Present Value
 - Payback period
 - Rough Order of Magnitude
- However, it is not always about the best return – organisations need to invest in all parts of their business

Module 3.5 – Investment Techniques – Return On Investment (ROI)

- ROI is income divided by investment
 - $\text{ROI} = (\text{total discounted benefits} - \text{total discounted costs}) / \text{total discounted costs}$
- The higher the ROI % or higher the ratio of benefits to costs, the better it is
- Many organisations have a required rate of return or minimum acceptable rate of return on investment for projects (11% to 14%)

Module 3.5 – Investment Techniques – Net Present Value (NPV)

- NPV is one of the most often used quantitative/financial models for project selection
- NPV is a method of calculating the expected net monetary gain or loss from an investment (project) by discounting all future costs and benefits to the present time
- Projects with a positive NPV should be considered if financial value is a key criterion
- Generally, the higher the NPV, the more favourable a project is

Module 3.5 – Investment Techniques – Payback period

- The payback period is the amount of time it takes a project before the accrued benefits surpass accrued costs or how much time an investment takes to recover its initial cost
- Based on tracking the net cash flow across each year to determine the year that net benefits overtake net costs (not discounted cash flows)
- Many organizations want IT projects to have a fairly short payback period (< 1 year) due to the changing nature of technology

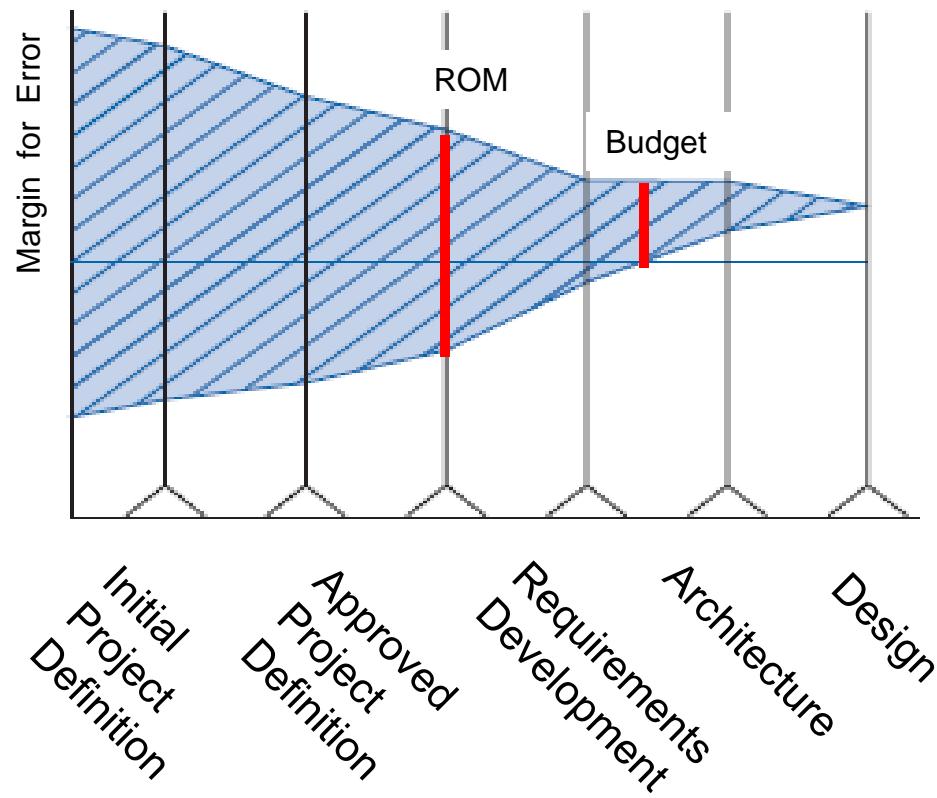


Module 3.5 – Investment Techniques – Project Estimation Rough Order of Magnitude (ROM)

The **Cone of Uncertainty** for cost estimates

Limited accuracy:

- ROM: -25% ... +75%
- Budget: -10% ... +25%



Reference: Kathy Schwalbe, *Information Technology Project Management*, pg 280

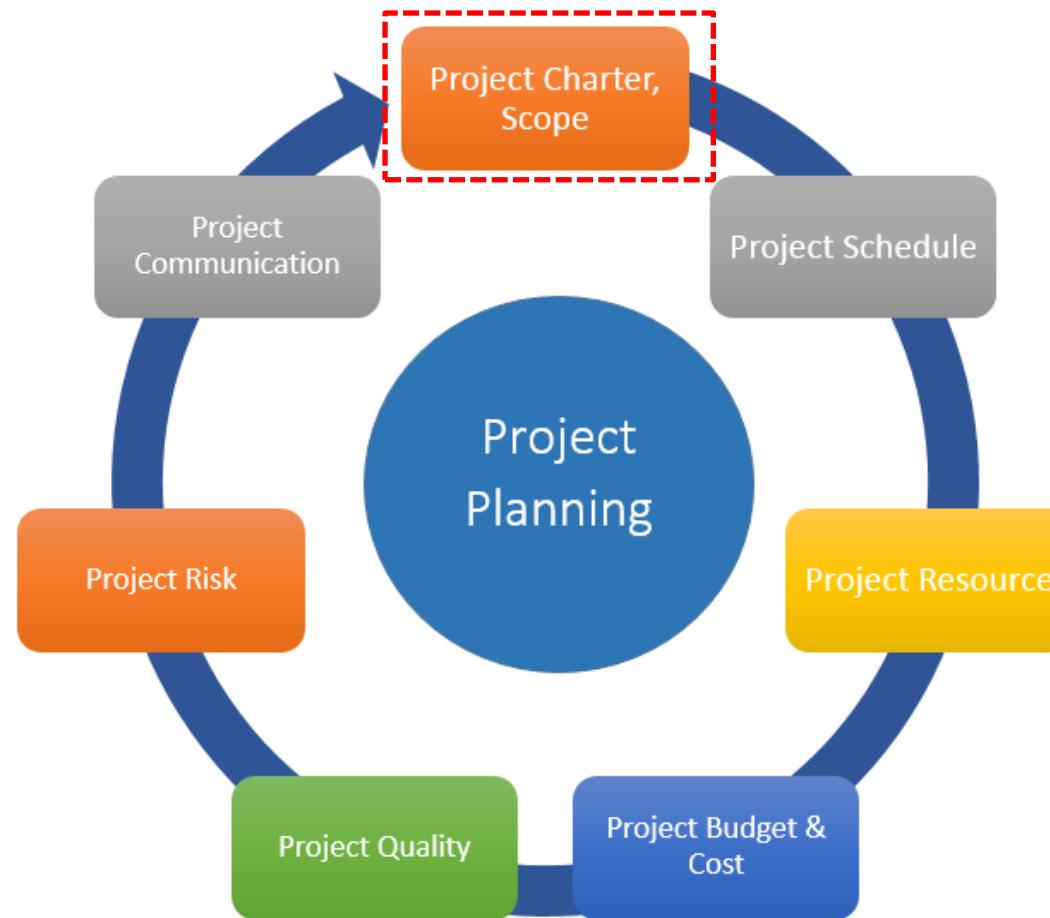
Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.



Module 3.6 – It all begins with a Project Charter



<http://blog.zilicus.com/software-project-management-activities-roles/>



Project Name

Target Date: [Date]

Project Description

Write out the project description here. Write out the project description here.

Costs	Item	Quantity	Rate	Total
	Resources			
	Equipment			
	Budget			
	Total			

Gains	Item	Quantity	Rate	Total
	Cost Savings			
	Time Savings			
	Revenue Gain			
	Net Total			

Project Team

- Person 1 – Project Manager
- Person 2 – Team Lead
- Person 3 – Analyst
- Person 4 – Developer
- Person 5 – Quality
- Person 6 – Trainer
- Person 7 – Other
- Person 8 – Other
- Person 9 – Other
- Person 10 – Other

Milestone 1

[Date]

[Description of what will be accomplished on this milestone]

Milestone 2

[Date]

[Description of what will be accomplished on this milestone]

Milestone 3

[Date]

[Description of what will be accomplished on this milestone]



Revision Poll

Lecture 1 – Intended Learning Objectives

Module 3: Projects

1. An initial look at (some) Project Management Methodologies / Standards.
2. Explore the key drivers of why projects fail / succeed.
3. Understand how organisations select the best / right projects (Project Screening).
4. Understand the Project Initialization process, Business Case structure and why organisations use them.
5. Explore various investment techniques and financial models.
6. Understand what a Project Charter is and how it is used.



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2021

2021 – Semester 1
Lecture 2

WILLING TO FAIL

UK Passport Agency (1999)

Passport fiasco cost ~ £13m (~ £6m on extra staff), with a £16,000 bill (umbrellas)+ 500 spoilt holidays

UK passport Agency -new computerised processing system to replace an old system and provide more secure passports.

They aimed to introduce the system in their Liverpool and Newport offices before the summer busy period.

[1]



What happened- The timeline was tight

18 November 1998 suspended the rollout of the computer system.

February 1999, applications began to increase and processing times rose.

March 1999, the delays started to attract Parliamentary and media attention. The number of applications then rose steeply.

[1]

Module 5. 1 - Identifying Stakeholders & the Stakeholder Register

Internal Stakeholders	External Stakeholders
Shareholders	End Users / Customers
Employees	Suppliers
Board Members	Governments
Sponsor / Business Managers	Unions
Project Manager	Local Communities / General Public
Management	Other Related Institutions
Project Team	Competitors

WILLING TO LEARN

What happened- Stakeholder communication

Should've consulted the stakeholder communication plan- put in measures to engage with the media and the public

Instead they withdrew some staff from the telephones and moved them to processing applications.

[1]

ANALYSIS OF CASE

Analysis

The timeline was tight

Agency failed to provide enough time for staff to learn and adapt to the new system.

Failure to adequately assess the time needed to implement the system and lack of a contingency plan in the event of a problem

[1]

Stakeholder analysis

Agency could have increased their channels of communication to engage with their key stakeholders

Instead they altered their recorded message to advise callers to write to the Agency and provided a fax number. The number of written queries quickly exceeded the Agency's ability to cope

[2]

WILLING TO FAIL

Stakeholder analysis

Without adequate information the feeling of panic increased and applicants were forced to join queues of thousands outside the passport offices.

The Agency responded by buying umbrellas and luncheon vouchers.

[2]



WILLING TO FAIL

Cost

£13m (~ £6m on extra staff)

£16,000 bill (umbrellas)

500 spoilt holidays

+ £9 million in lost business borne by Siemens Business Services

[1]

References

[1] (05/03/2021) The Passport Delays of Summer 1999. [Online]

<https://www.nao.org.uk/report/united-kingdom-passport-agency-the-passport-delays-of-summer-1999/>

[2] (05/03/2021) Why Stakeholder Management is so important. [Online]

<https://www.stakeholdermap.com/importance-stakeholder-management.html>



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 2

Lecture 1 – Recap

- ✓ Understand Assignments and our expectations
- ✓ Understand key elements of a Project and why organisations use them
- ✓ Understand the foundational components of Project Management
- ✓ Understand key skills, responsibilities & activities of a Project Manager
- ✓ Understand key elements of how to manage Projects
- ✓ Exposure to some Project Management Methodologies

Lecture 1 – Recap

- ✓ Explore key drivers in why projects fail / succeed
- ✓ Understand how organisations select the best / right projects
- ✓ Understand the Project Initialization process, Business Case structure and why organisations use them
- ✓ Explore various Investment techniques and financial models
- ✓ Understand responsibilities associated with building a Business Case and the accountable group / individual
- ✓ Understand what a Project Charter is and how it is used



WILLING TO LEARN

L1 - Recap



Intended Learning Objectives

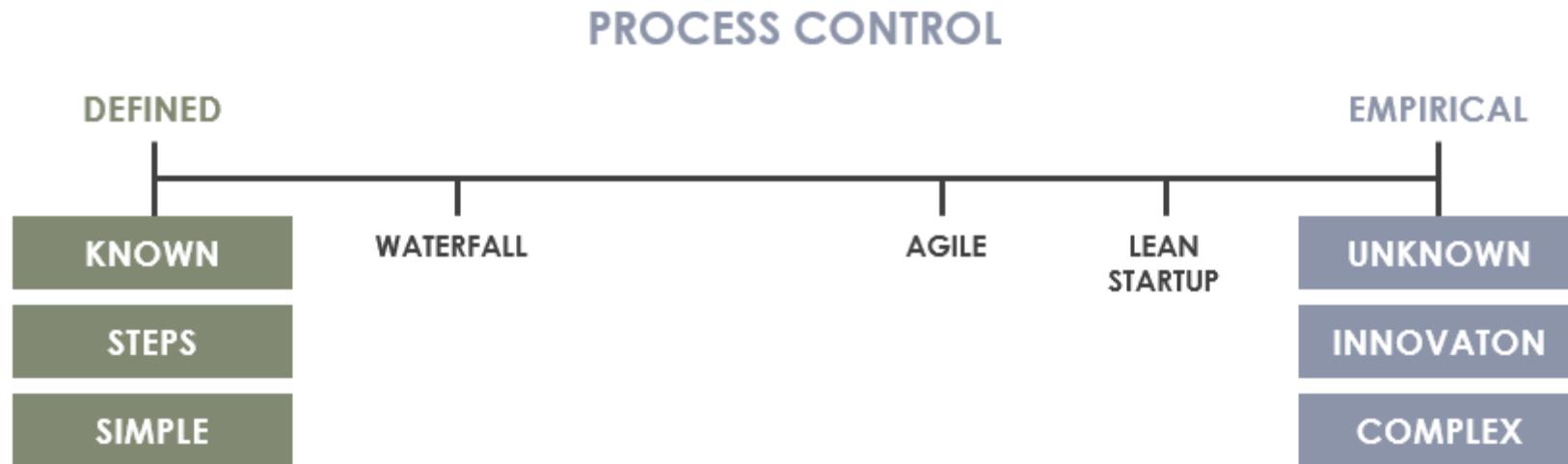
Module 4 – Process & Project Management Plan.

Module 5 – Stakeholder Management.

Module 6 – Communication Management.

Module 4.1 – Empirical and Defined Process

Empirical process control expects the unexpected, while defined process control expect every piece of work to be completely understood in upfront.

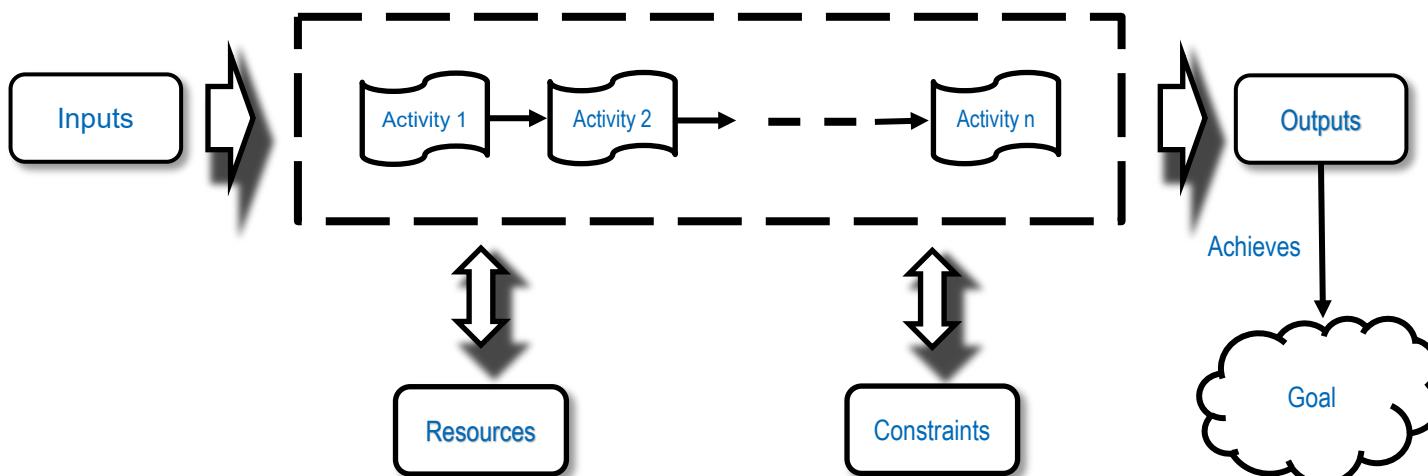




Module 4.1 – Defined Process Control

A process with a well-defined set of steps. Given the same inputs, a defined process should produce the same output every time.

Great when in an environment with relatively low volatility that can be easily predicted; given the same inputs, a defined process should produce the same output every time based on its repeatability and predictability nature.



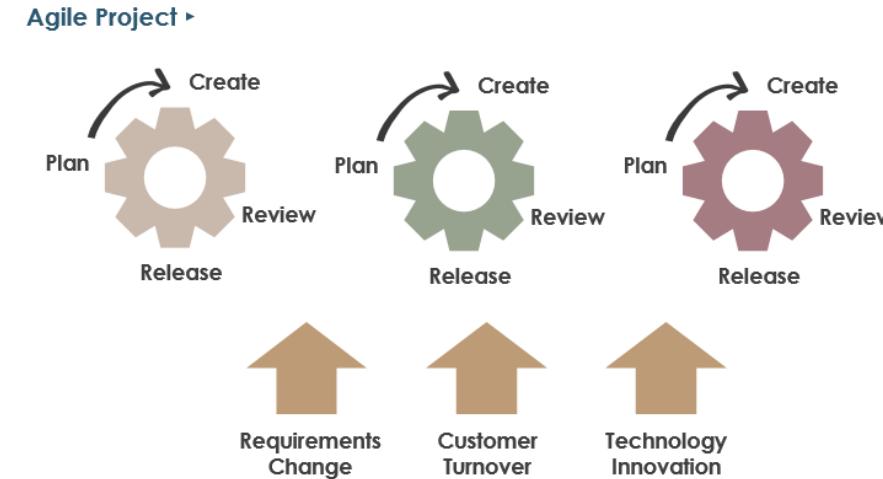
www.visual-paradigm.com/scrum/empirical-vs-defined-process-control



Module 4.1 – Empirical Process Control

In empirical process control, you expect the unexpected. Empirical process control has the following characteristics:

- Learn as we progress
- Expect and embrace change
- Inspect and adapt using short development cycles
- Estimates are indicative only and may not be accurate





WILLING TO LEARN

Module 4.1 – Defined v Empirical



Defined / Repeatable Process



www.mountaingoatsoftware.com/exclusive/scrum-foundations

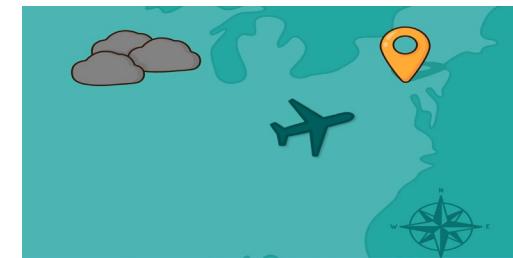
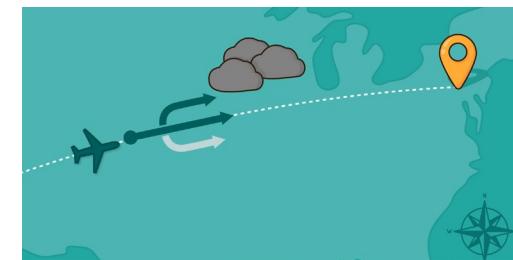


WILLING TO LEARN

Module 4.1 – Defined v Empirical



Defined / Repeatable Process





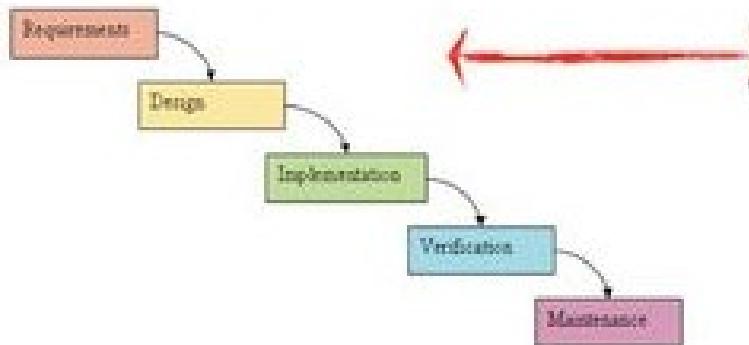
Module 4.1 – Defined v Empirical



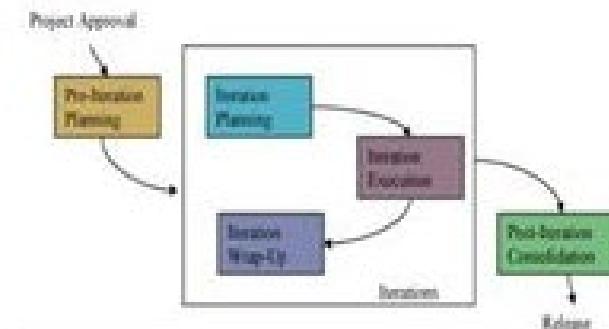
Requires that *every piece of work be completely understood*. Given a well-defined set of inputs, the same outputs are generated every time. A defined process can be started and allowed to run until completion, with the same results every time.

Provides and exercises control through transparency, frequent inspection and adaptation for processes that are imperfectly defined and generate unpredictable and unrepeatable outputs.

Waterfall



Agile



Module 4.1 – What does a Process have to do with Project Management and Software Engineering?

1. Project Management is a process as it defines a series of tasks (Planning, Executing and Controlling) to deliver a specific / an agreed set of outcomes.
2. System Development Lifecycle (SDLC) is a term used in Software Engineering. It describes a process for planning, creating, testing, and deploying an information system. SDLC can be composed of hardware only, software only, or a combination of both.

Module 4.1 – Project Management Plan (Formal)

Almost every organisations will have it's own “*version*” of a Project Management Plan (PMP), however the reasons they have and use them are the same.

A PMP is a formal approved document that defines how the project is executed, monitored and controlled. It may be a summary or a detailed document.

It is a document that is owned, controlled and populated by the Project Manager and is used throughout the project.

A good PMP provides the required level of detail across key project components and is the one source of truth for all parties involved across the project.



MILLENNIUM

Module 4.1 – Project Charter V Project Management Plan

A Project Charter is a summary project proposal to secure approval for the project goals and terms (useful as part of Business Case).

A PMP is an approved document showing how to achieve the approved project goals / benefits and provides the details on how to execute and manage the project (used as part of mobilisation and on-going management of the project).

 THE UNIVERSITY OF MELBOURNE

Simple Project Charter

Project Name: [REDACTED]

Target Date: [Date]

Project Description:
Write out the project description here. Write out the project description here.

Costs

Item	Quantity	Rate	Total
Resources			
Equipment			
Budget			
Total			

Gains

Item	Quantity	Rate	Total
Cost Savings			
Time Savings			
Revenue Gain			
Net Total			

Project Team

- Person 1 – Project Manager
- Person 2 – Team Lead
- Person 3 – Analyst
- Person 4 – Developer
- Person 5 – Quality
- Person 6 – Trainer
- Person 7 – Other
- Person 8 – Other
- Person 9 – Other
- Person 10 – Other

Milestone 1 [date]
[Description of what will be accomplished on this milestone]

Milestone 2 [date]
[Description of what will be accomplished on this milestone]

Milestone 3 [date]
[Description of what will be accomplished on this milestone]

SWEN90016 Software Processes and Project Management -33- IT ALL STARTS HERE

Primary Use: Summary (few pages) of key information used to communicate, engage, gain buy-in and obtain approvals.

Project Management Plan - Template			
1. Title Page			
2. Executive Summary			
3. Table of Contents			
4. Introduction			
5. Project Scope			
6. Project Plan			
7. Team			
8. Estimates			
9. Risks			
10. Stakeholders			
11. Appendix			
Section 1 - Introduction			
1.1 Purpose			
The purpose of this guideline is to assist the contractor and project team in creating a project specific Design Quality Control Plan for incorporation activities. These guidelines represent minimum requirements for plan implementation. The Design Consultant is responsible for quality merging these guidelines with corporate Quality Assurance/Quality Control requirements. The Design Consultant is responsible for the overall quality of the final design product and is required to incorporate the review of all risk conditions into the project specific Design Quality Control.			
A well structured Design Quality Control Plan helps to ensure that designs are economical, cost effective, safe and appropriate for their location and usage. The project team is required to submit the project specific Design Quality Control Plan to the UQDC Quality Control.			
UQDC Quality Control reviews the project specific Design Quality Control Plan prior to the Consultant starting an Project Work. Adherence to the project specific Design Quality Control Plan is required signature of a checklist or affidavit.			
The project specific Design Quality Control Plan will be reviewed that the design and project documents are consistent with the design and project documents, available techniques, available resources and reasonable decided by competent professionals. The project specific Design Quality Control Plan is a practical tool and does not replace the sound judgment and experience of competent professionals. The project specific Design Quality Control Plan is responsible for the quality of project specific Design Quality Control.			
UQDC reserves the right to audit the Design Consultant for compliance of Quality Assurance/Quality Control procedures as provided in the Major Professional Services Agreement. No additional compensation is provided to the Design Consultant for time or materials associated with the audit.			
12. Requirements Analysis			
13. Requirements Gathering and Definition			
14. Requirements Specification			
15. Requirements Verification			
16. Requirements Change Control			
17. Requirements Analysis			
18. Requirements Gathering and Definition			
19. Requirements Specification			
20. Requirements Verification			
21. Requirements Change Control			
22. Requirements Analysis			
23. Requirements Gathering and Definition			
24. Requirements Specification			
25. Requirements Verification			
26. Requirements Change Control			
27. Requirements Analysis			
28. Requirements Gathering and Definition			
29. Requirements Specification			
30. Requirements Verification			
31. Requirements Change Control			
32. Design			
33. Design Verification			
34. Design Change Control			
35. Complete Draft of Design			
36. Complete Update of Design			

Primary Use: Detailed document used to establish and manage the project. Defines all key items the project needs to consider.

Module 4.1 – Project Management Plan (Formal)

A typical PMP consists of all / or most of the following categories.

- *Project Information*
 - Executive Summary
 - Financial Authority to proceed
 - Key Stakeholders
 - Scope
 - Delivery approach / SDLC - Waterfall or Agile
 - Resources / People
 - Key Milestones
 - Project Budget
 - Lessons learned applied to this project
 - Constraints

WILLING TO LEARN

Module 4.1 – Project Management Plan (Formal)

And.....

- *Project Governance*
 - Roles and Responsibilities
 - Mandatory Project Planning / Key Additional Activities
 - Schedule
 - Risk Management
 - Cost Estimation
 - Quality Assurance
 - Configuration Management (Change Management)

The PMP is a large multi-page document that takes time to prepare, review and complete. Multiple people (subject experts) are involved and prepare the specific details. The Project Manager coordinates all items and has ultimate accountability for the quality and final outcome.



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 2

Intended Learning Objectives

Module 4 – Process & Project Management Plan.

Module 5 – Stakeholder Management.

Module 6 – Communication Management.

Intended Learning Objectives

Module 5 – Stakeholder Management.

1. Stakeholders & the Stakeholder Register.
2. Stakeholder Engagement and Planning.

Module 5. 1 - Identifying Stakeholders & the Stakeholder Register

Internal Stakeholders	External Stakeholders
Shareholders	End Users / Customers
Employees	Suppliers
Board Members	Governments
Sponsor / Business Managers	Unions
Project Manager	Local Communities / General Public
Management	Other Related Institutions
Project Team	Competitors



WILLING TO LEARN

Module 5. 1 - Identifying Stakeholders & the Stakeholder Register

Name	Position	Internal/External	Project Role	Contact Information
Stephen	VP of Operations	Internal	Project Sponsor	stephen@globaloil.com
Betsy	CFO	Internal	Senior Manager. Approves Funds	betsy@globaloil.com
Chien	CIO	Internal	Senior Manager. PM's Boss	chien@globaloil.com
Ryan	IT Analyst	Internal	Team Member	ryan@globaloil.com
Lori	Director Accounting	Internal	Senior Manager	lori@globaloil.com
Sanjay	Director Refineries	Internal	Senior Manager of Largest Refinery	sanjay@globaloil.com
Debra	Consultant	External	Project Manager	debra@globaloil.com
Suppliers	Suppliers	External	Software Supplier	suppliers@gmail.com

Intended Learning Objectives

Module 5 – Stakeholder Management.

1. ~~Stakeholders & the Stakeholder Register.~~
2. Stakeholder Engagement and Planning.

Module 5. 2 – Understanding Stakeholder Engagement and Planning

Levels of Stakeholder Engagement

- *Unaware*: Unaware of the project and its potential impacts on them
- *Resistant*: Aware of the project yet resistant to change
- *Neutral*: Aware of the project yet neither supportive nor resistant
- *Supportive*: Aware of the project and supportive of change
- *Champion / Leading*: Aware of the project and drives change

Module 5. 2 – Understanding Stakeholder Engagement and Planning

The stakeholder management plan can include:

- Current and desired engagement levels
- Interrelationships between stakeholders
- Communication requirements
- Potential management strategies for each stakeholder
- Methods for updating the stakeholder management plan

Note: Because a stakeholder management plan often includes sensitive information, it may not be part of the official project documents, which are normally available for all stakeholders to review.

In many cases, only project managers and a few other team members should prepare the stakeholder management plan. Parts of the stakeholder management plan may not be written down, and if they are, distribution is limited.

Module 5. 2 – Understanding Stakeholder Engagement and Planning

Stakeholder Analysis includes:

- Names and Organisations of Key Stakeholders
- Their Role on the Project
- Unique Facts about Each Stakeholder
- Level of Interest in the Project
- Influence on the Project
- Suggestions and Strategies for Managing Relationships with each Stakeholder



Module 5. 2 – Understanding Stakeholder Engagement and Planning

Stakeholder Analysis example:

Name	Power / Influence	Current Engagement	Potential Management Strategies
Brian	High/High	Champion / Leading	Brian can seem intimidating due to his physical stature and deep voice, but he has a great personality and sense of humour. He previously led a similar software upgrade project at another company and knows what he wants. Manage closely and ask for his advice as required. He likes to be kept in touch with short, frequent updates in person.
Mary	High/Medium	Resistant	Mary is very organised yet hardhead. She has been pushing corporate IT standards, and the system the PM and sponsors like best goes against those standards, even though it's the best solution for this project and the company as a whole. Need to convince her that it is ok and that people still respect her work and position.
Finance Team	Medium/High	Resistant	The Finance Team is resistant to the Project. They believe the funds can be better used elsewhere in the organisations. They also believe the Benefits are not achievable in the defined pay back period. Key stakeholder as they control the funding which the project is dependant on. Need to convince them of the costs & benefits and ensure they understand the detail. Get them involved in a detailed review of all \$'s.
Jessica	High/Low	Neutral	Very professional, logical person. Gets along with Mary. She has supported Brian in approving past projects with strong business cases. Provide detailed financial justification for the suggested solution to keep her satisfied. Also ask her to talk to Mary on Brian's behalf.



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 2

Intended Learning Objectives

Module 4 – Process & Project Management Plan.

Module 5 – Stakeholder Management.

Module 6 – Communication Management.

Intended Learning Objectives

Module 6 – Communication Management.

1. The communication challenge.
2. The importance of listening.
3. Communication key skills & importance.
4. Communication plans.
5. Virtual teams & communication.
6. Key communication considerations.

Module 6. 1 – Communication Challenges



Individual



Team

- Semantics [meaning]
- Perception [interpretation]
- Communication Channel
- Feedback
- Anxiety
- Culture

- Status
- Silos
- Information Overload
- Lack of Communication
- Protocol [rules]



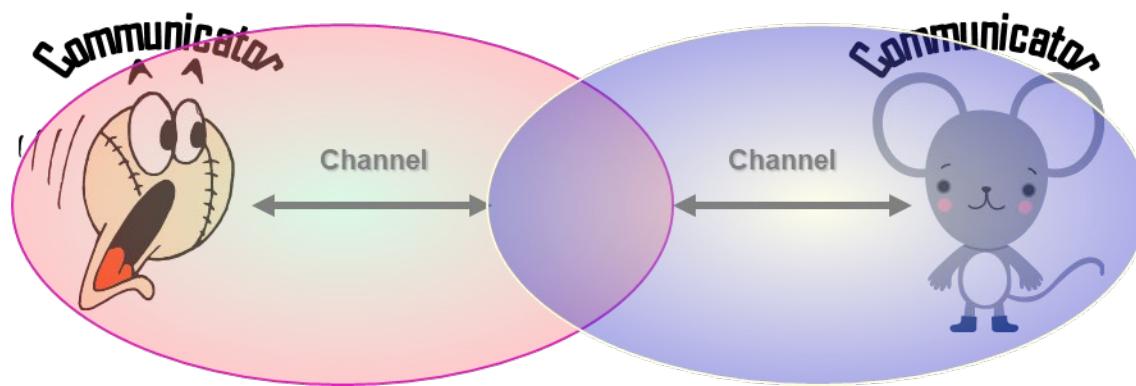
WILLING TO LEARN

Module 6. 1 – Communication Challenges



The Communication Model

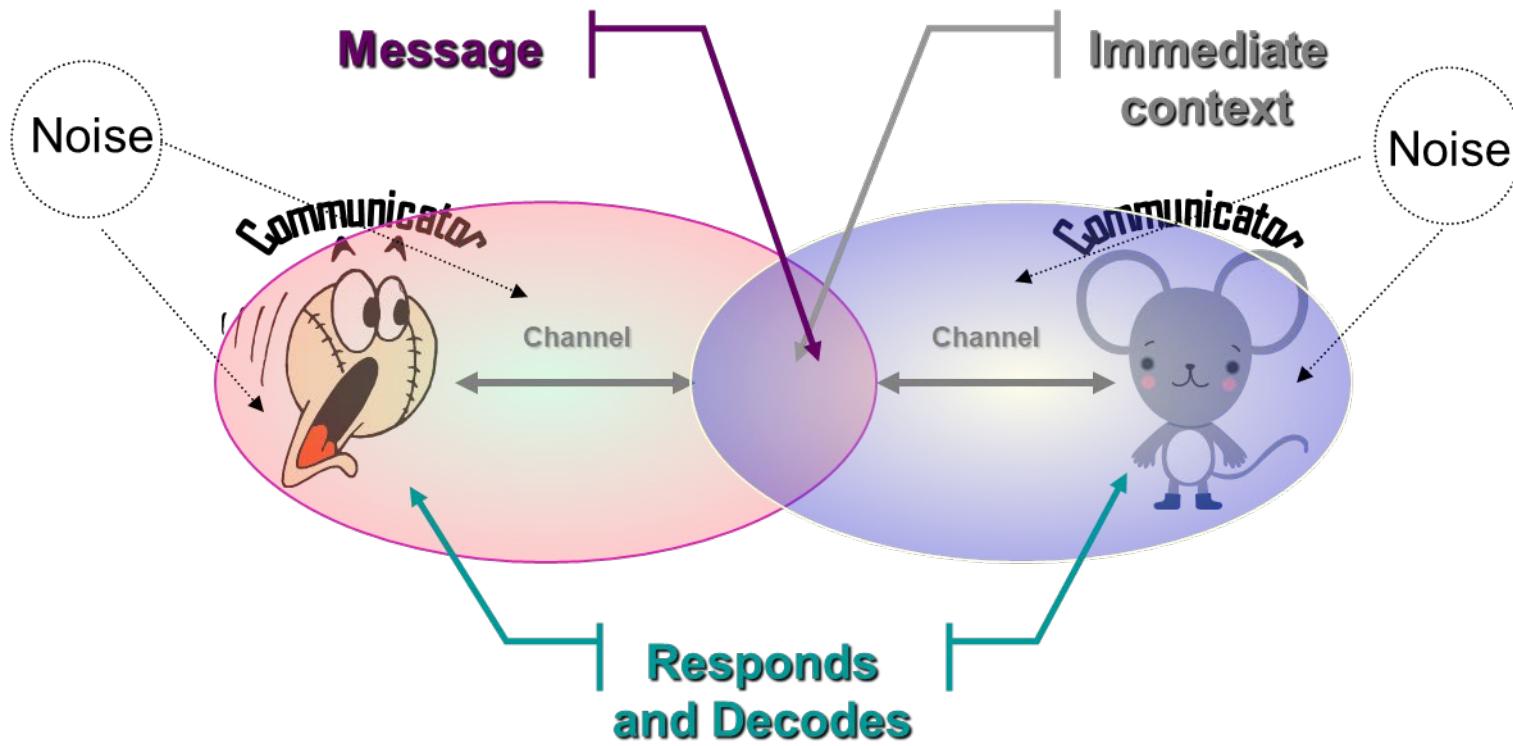
Module 6. 1 – Communication Challenges



The Communication Model



Module 6. 1 – Communication Challenges



The Communication Model

Intended Learning Objectives

Module 6 – Communication Management.

1. ~~The communication challenge.~~
2. **The importance of listening.**
3. Communication key skills & importance.
4. Communication plans.
5. Virtual teams & communication.
6. Key communication considerations.

Module 6. 2 – The importance of Listening

Hearing is the act of perceiving sound by the ear.

Listening requires concentration and is the process of taking in what you hear and mentally organising it so it makes sense.

**The Act of Listening
Demands Real Effort**

**Listening is An
Essential Life Skill**



**Truly Effective
Listeners are Rare**

**Few People Practice
Listening and Even Fewer
Have Been Trained to Listen**

Module 6. 2 – The importance of Listening

Why Do We Listen?

- Promotes problem-solving abilities
- Demonstrates acceptance of others
- Builds and retains trust in relationships
- Increases speaker's receptiveness to thoughts and ideas of others
- Increases self-esteem of the speaker helps you evaluate messages
- Helps you understand and retain information
- Allows you to help others

Module 6. 2 – The importance of Listening

The Process of Listening

- Predicting [some expected outcome]
- Receiving
- Assigning meaning
- Assess / Validate
- Remembering

Module 6. 2 – The importance of Listening

- Types of Listening
 - Passive Listening – *Lectures*
 - Taking in the information without processing or reacting
 - Active or Empathetic Listening – *Tutorials*
 - Show interest
 - Asks questions
 - Avoid distractions
 - Use direct eye contact
 - Do not interrupt
 - Read both verbal and nonverbal messages

Module 6. 2 – The importance of Listening

Challenges to Listening

- Physiological limitations
- Inadequate background information
- Selective memory or expectations
- Fear of being influenced / persuaded
- Bias and being judgemental
- Boredom or interference from emotions
- Partial listening and distractions e.g. mobile phones / background noise
- Physical barrier e.g. environment, lighting, uncomfortable seating
- Cultural differences [understanding the spoken words]
- Past experiences
- Jargon & Acronyms

Module 6. 2 – The importance of Listening

The Importance of Active Listening

- Shows the speaker you are concerned or interested
- Leads to getting better information
- Encourages further communication
- Has the potential to enhance relationships
- Can calm down someone who is upset
- Invites others to listen to you
- Leads to better co-operation and problem solving

Module 6. 2 – The importance of Listening

TED Talk: 5 ways to listen better

Julian Treasure

7 min video

Intended Learning Objectives

Module 6 – Communication Management.

1. ~~The communication challenge.~~
2. ~~The importance of listening.~~
3. **Communication key skills & importance.**
4. Communication plans.
5. Virtual teams & communication.
6. Key communication considerations.



WILLING TO LEARN

Module 6. 3 – Communication key skills & importance

Communication Skills are critical in Project Management

- Conveying your point of view
- Motivating and influencing others
- Delegating
- Recognising, defining and solving problems
- Delivering presentations / updates
- Setting goals & articulating a vision
- Managing conflict
- Networking
- Negotiating

L3.1 – Teams, Individuals and Motivation

So why is this important to Project Management?

L3.2 – Project Manager Key Activities "a change is occurring"

Agile is redefining the way we execute projects and the role of the PM. In Agile:

- No defined PM role
- Many roles and tasks are shared across team members
- Key project activities are self-understanding formal and appropriate documentation
- Some alignment to activities of a Scrum Master
- Major focus on communication and collaboration between stakeholders
 - Increases and facilitates teams to communicate
 - Managing and structuring scarce resources
 - Managing risk
 - Identifying and clearing issues
 - Managing and implementing change
 - Retaining and re-using knowledge
 - Organisational wide learning from past success and failures
 - Allows the team to self-organise and work more effectively
 - Assists others with key issues

Module 6.3 – Communication key skills & importance

Why Is This Important?

Because successful Project Managers MUST have the ability to:

- Read / understand the client
- Run a meeting
- Communicate (written & orally) thoughts accurately
- Manage the team
- Influence your environment
- Ensure alignment and buy-in to the purpose / outcome

Intended Learning Objectives

Module 6 – Communication Management.

1. ~~The communication challenge.~~
2. ~~The importance of listening.~~
3. ~~Communication key skills & importance.~~
4. **Communication plans.**
5. Virtual teams & communication.
6. Key communication considerations.

Module 6. 4 – Communication Plan

- A large proportion of a Project Managers time is spent on communication
- Project Managers often use a Communications Plan to assist in managing and coordinating key communication messages
- A good project Communication Plan:
 - Ensures communications is effective and efficient
 - Allows the Project Manager to be pro-active
 - Sets a common understand of what will be done and when
 - Clarifies who is responsible for key items, what will be delivered and by who

<http://www.projectmanagementdocs.com/project-planning-templates/communications-management-plan.html#ixzz5A9VcGljd>

Module 6. 4 – Communication Plan

A Communications Plan defines:

- What information will be communicated - detail and format
- Communication Channel - meetings, email, telephone, web portal, etc.
- When information will be distributed – frequency of formal and informal comms
- Who is responsible
- Communication needs of stakeholders
- Resources the project will allocate for communication
- How sensitive or confidential information will be communicated & who will authorise this
- The flow of project communications
- Any constraints (internal or external) which affect project communications
- Any standard templates, formats, or documents the project must use
- Escalation process for resolving any communication-based conflicts or issues

Module 6. 4 – Communication Plan

Example

Is underpinned by a Communications Matrix

Stakeholder	Communication Objective	Format	Frequency	Owner	Importance
Sponsor	Provide updates on project progress, key issues, success and support required	Regular Meeting - face to face Formal Report	Weekly Monthly	Project Manager	High
Business Expert	Gather requirements, sign-off all scope, approve prototype and final acceptance	Formal Report / documentation	Fortnightly	Project Manager	High
Finance	Future funding approval	Project Finances	Bi-monthly	Finance rep	High
Human Resources	Identify staff required for project and deal with all staff related items	Resource plans	Monthly	Project Support	Medium
Risk Department	Identify risks and mitigation strategies and ensure they are being followed	Risk Management Plan	Monthly	Project Support	Medium
Internal IT Staff	Identify resources for all phases including Design, Requirements Gathering, Development and Production Implementation	Regular Meeting - face to face Formal Report	Weekly Monthly	Project Manager	High
External IT Staff / Supplier	To ensure they can execute on their Testing Services contract	Formal Report	Monthly	Project Manager	Low

Intended Learning Objectives

Module 6 – Communication Management.

1. ~~The communication challenge.~~
2. ~~The importance of listening.~~
3. ~~Communication key skills & importance.~~
4. ~~Communication plans.~~
5. **Virtual teams & communication.**
6. Key communication considerations.

Module 6. 5 – Virtual Teams & Communication

What is a virtual team?

- A virtual team (also known as a geographically dispersed team, distributed team, or remote team) usually refers to a group of individuals who work together from different geographic locations and rely on communication technology. (Wikipedia)

Module 6. 5 – Virtual Teams & Communication

In 2014 survey 1,700 knowledge workers, 79% reported working always or frequently in dispersed teams. Armed with laptops, Wi-Fi, and mobile phones, i.e. technology most professionals can do their jobs from anywhere.

Why does it appeal to employees?

- Employees can be more flexible with work and home commitments

Why does it appeal to organisations?

- Organisations can access the best GLOBAL talent
- Save on real estate costs

Module 6. 5 – Virtual Teams & Communication

What does this mean for communication

- Communication is less rich and less frequent than face-to-face interaction

Why?

- Less visual and behavioural cues
- Less or no informal interactions

BUT not all bad

- Those less inclined to speak in groups, may feel more comfortable
- Less importance on interpersonal skills and physical appearance may benefit certain members of the team
- Still need to be mindful of unconscious bias (virtual unconscious bias)

Module 6. 5 – Virtual Teams & Communication

Create a Communication charter.

- Discipline about how the team should communicate
- Norms of behaviour when participating in virtual meetings (background noise, side conversations, talking clearly and at a reasonable pace, listening attentively, not dominating the conversation)
- Guidelines on communication modes - in which circumstances, which mode should be used e.g. email should be used for formal correspondence, a WhatsApp group for chatting informally, documents

Module 6. 5 – Virtual Teams & Communication

Leverage communication technologies

- How can you use technology to function effectively as a team?
- Who has access to which technologies - bandwidth, free vs cost.

Team building

- Virtual water cooler
- Formal and informal team building activities
- For instance we have a random channel in our slack group where we post funny jokes, memes, celebrations e.g. birthdays, milestones. An intentional place to strengthen the group
- Think of how you can use social networking tools and features to create a connected team.

Module 6. 5 – Virtual Teams & Communication

Factors that contribute to a good virtual team

- Good communication skills
- High emotional intelligence
- Ability to work independently
- Resilience
- Awareness and sensitivity to other cultures is important especially in global groups

Guess what – They are very similar to factors that contribute to any successful team!

References:

<https://hbr.org/2013/06/making-virtual-teams-work-ten>

<https://hbr.org/2014/12/getting-virtual-teams-right>

<https://hbr.org/2018/02/how-to-collaborate-effectively-if-your-team-is-remote>

Intended Learning Objectives

Module 6 – Communication Management.

1. ~~The communication challenge.~~
2. ~~The importance of listening.~~
3. ~~Communication key skills & importance.~~
4. ~~Communication plans.~~
5. ~~Virtual teams & communication.~~
6. **Key communication considerations.**

Module 6. 6 – Key Communication Consideration

Importance of Face to Face meetings

- 58% of communication is through body language
- 35% of communication is through how the words are said
- 7% of communication is through content or words that are spoken
- Pay attention to more than just the actual words
- A person's tone or voice and body language can say a lot about how they really feel
- Non verbal comms govern how other people feel about us and also how we feel about ourselves

Body Language - Spend some time looking at Ted Talk https://www.youtube.com/watch?v=Ks-_Mh1QhMc

Module 6. 6 – Key Communication Consideration

Items to remember

- Rarely does the receiver interpret a message exactly as the sender intended
- Geographical location and cultural background affect the complexity of communications
 - Different working hours
 - Language barriers
 - Different cultural norms
- Communication helps manage conflicts effectively
- Spend time developing communication skills – *practice & feedback*
- Choose the channel appropriately



WILLING TO LEARN

Module 6. 6 – Key Communication Consideration Communication Channels – some examples

Choose carefully as it will make a difference

How well medium is Suited to:	Hard Copy	Telephone Call	Voice Mail	eMail	Meeting / f2f	Web Site
Confirming commitments	1	3	3	1	2	1
Building consensus	3	2	3	3	1	3
Mediating conflict	3	2	3	3	1	3
Resolving misunderstanding	3	1	3	3	1	3
Addressing negative behaviour	3	2	3	2	1	3
Expressing support / appreciation	1	1	2	1	1	1
Encouraging creative thinking	2	3	3	2	1	3

1 = Most suited, 2 = Less suited and 3 = Least Suited

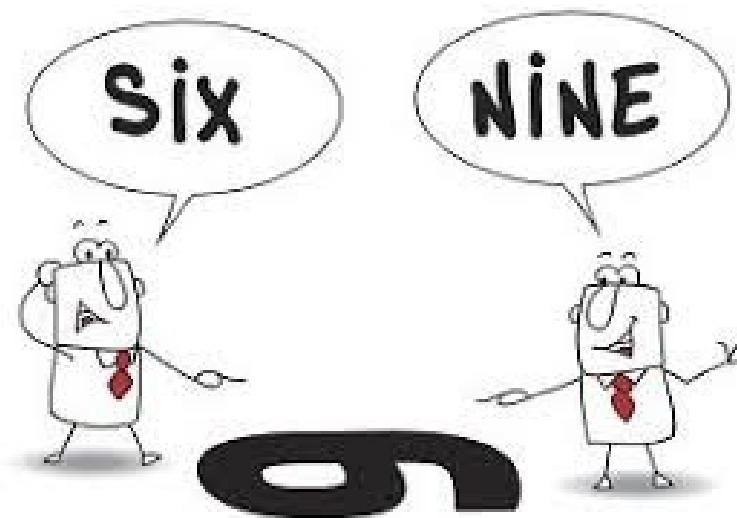
Module 6. 6 – Key Communication Consideration Conflict

Conflict is the single most undermanaged activity in projects and if left unresolved will destroy a project. Key causes include:

- Schedule
- Intellectual disagreements
- Personalities
- Project Priorities
- Manpower
- Technical
- Administration
- Personality
- Cost

Module 6. 6 – Key Communication Consideration Conflict

Deal with it quickly and succinctly and
BEFORE it becomes a major issue





SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 3

Lecture 3 – Intended Learning Objectives

**Module 7 – Software Development
Lifecycles - Formal.**

**Module 8 – Software Development
Lifecycles - Agile.**

Lecture 3 – Intended Learning Objectives

Module 8 - Software Development Lifecycles

Agile:

1. Understand what Agile is and its origins.
2. Understand the Agile framework.
3. Understand Scrum – Roles, Ceremonies and Artefacts.
4. Understand advantages / disadvantages of Agile.



Module 8.1 – SDLCs

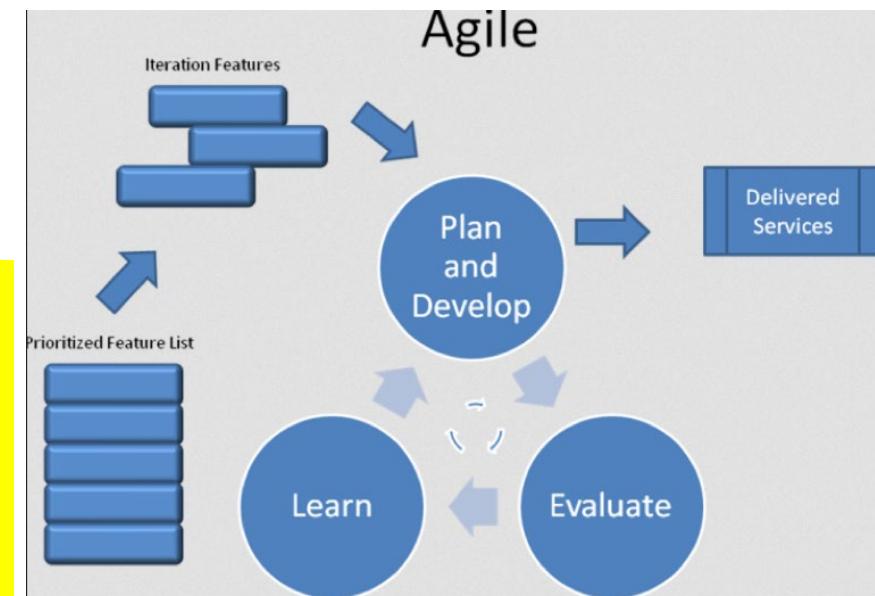
There are many SDLCs around with organisations typically favouring a blend of Formal and Agile approaches.

1. Formal

- Waterfall
- Incremental
- V-Model

2. Agile

- Kanban
- Scrum
- Extreme Programming



Module 8.1 – Why is Agile attractive

- We are in an ever changing global world with the pace of change increasing
- Customer needs and demands are exponentially increasing – products must continually be delivered
- Low Technology cost, ease of use and the global market place has increased competition and reduced entry barriers
- The war for talent is over – and we have lost! Cross functional teams help minimise the potential loss
- Long development cycles are like long lunches – a thing of the past
- Quality is no longer something we do / check later – it must be part of everything we do
- Cross functional groups are more fun!

Module 8.1 – What is Agile

- A framework based on iterative development where requirements and solutions evolve through collaboration between self-organising cross-functional teams
- A disciplined process that encourages frequent inspection and adaptation
- A leadership philosophy that encourages teamwork, self-organisation and accountability
- A set of engineering best practices intended to allow for rapid delivery of high-quality software
- A business approach that aligns development with customer needs and company goals

Module 8.1 – What is Agile

- In software development, we think about methodologies, activities, interactions, results, work products, artefacts and processes to organise the work.

The main tasks of software development remain the same regardless of methodology used, however with Agile, the flow of activities, how they are undertaken and who is involved is extremely different.

Module 8.1 – What is Agile - Origins

- Changes initiated in large US corporates in 1990's
- Software engineers frustrated with
 - long lead times before products were delivered
 - Decisions made early in the project couldn't be changed later
- 17 software engineer thought leaders met first in 2000 to discuss software engineering and different approaches
- Famous meeting in 2001 at the Snowbird ski lodge in Utah where they met to change the way the industry designed, engineered and deployed software
- Brought together by the Agile Manifesto

<https://techbeacon.com/agility-beyond-history%E2%80%94-legacy%E2%80%94-agile-development>

Lecture 3 – Intended Learning Objectives

Module 8 - Software Development Lifecycles

Agile:

1. ~~Understand what Agile is and its origins.~~
2. Understand the Agile framework.
3. Understand Scrum – Roles, Ceremonies and Artefacts.
4. Understand advantages / disadvantages of Agile.

Module 8.2 – Agile Framework

Primary elements of the Agile framework include:

- Manifesto
- 12 Key Principles
- Kanban
- Scrum

Module 8.2 – Agile Framework - *Manifesto*

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- | | | |
|--------------------------------|------|-----------------------------|
| ■ Individuals and interactions | over | processes and tools |
| ■ Working software | over | comprehensive documentation |
| ■ Customer collaboration | over | contract negotiation |
| ■ Responding to change | over | following a plan |

That is, while there is value in the items on the *right*, we value the items on the *left* more.

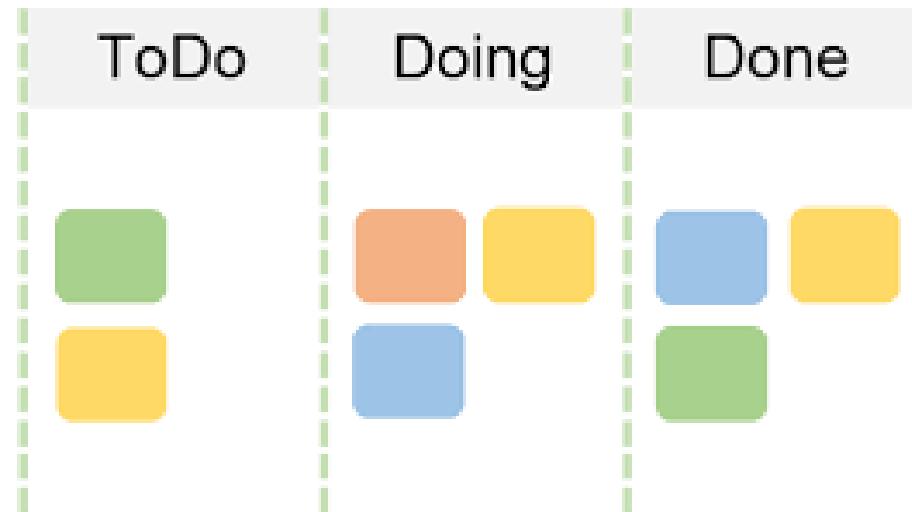
<http://www.agilealliance.org>

Module 8.2 – Agile Framework – *12 Key Principles*

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, shorter timeframes is the preference.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done - is essential.
11. The best architectures, requirements, and designs emerge from self-organising teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Module 8.2 – Agile Framework - *Kanban*

- ***Signboard / Billboard:*** Work items are visualised to provide participants a view of progress and process, from start to finish usually via a Kanban board

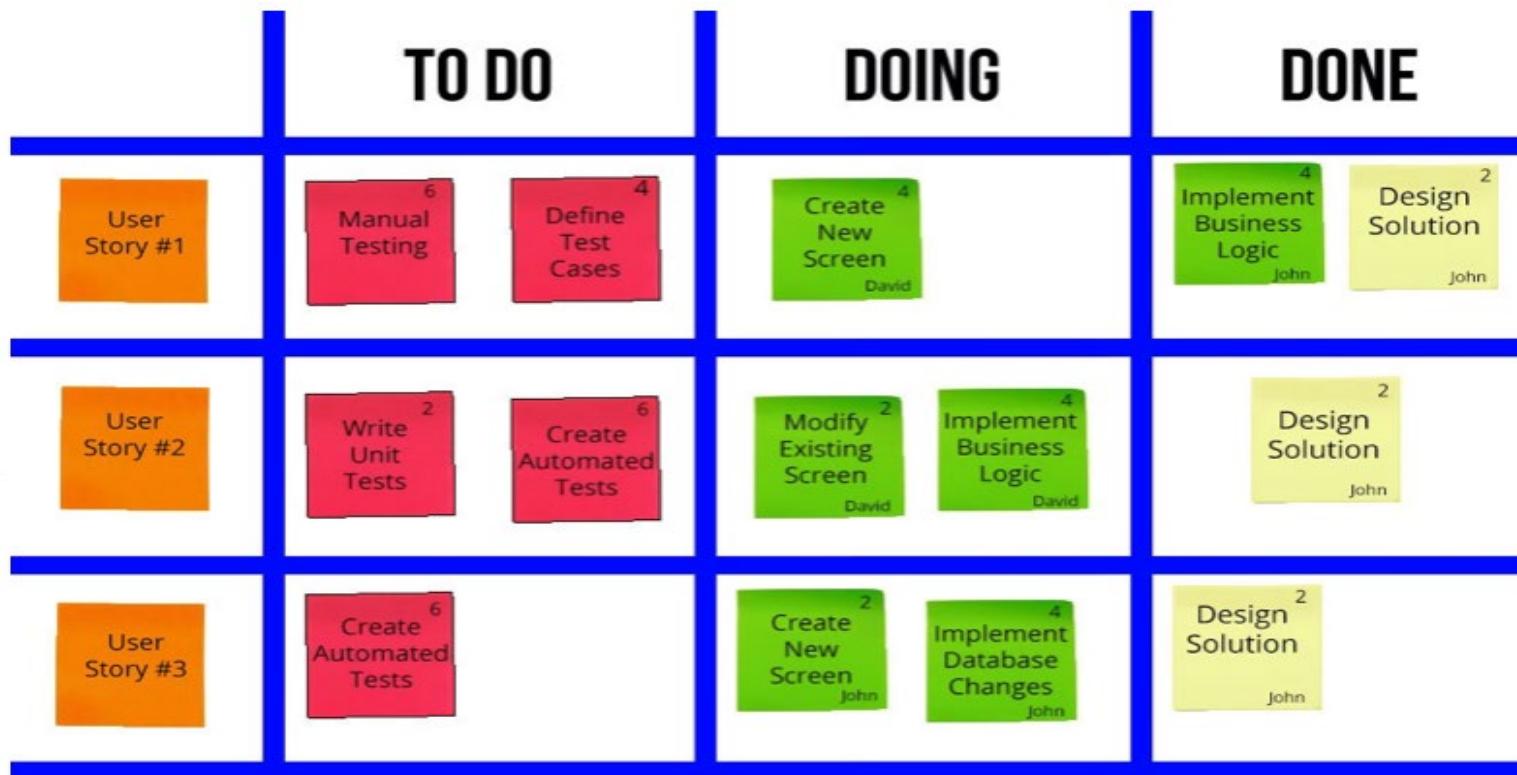




SWEN90016

Module 8.2 – Agile Framework - *Kanban*

Visual progress gives transparency/accountability for self-organizing teams often referred to as **SWIMLANE** boards

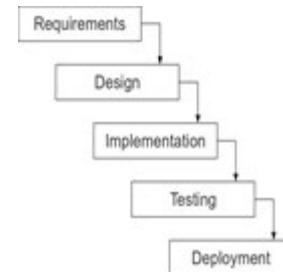


Module 8.2 – Agile Framework - *Scrum*

Scrum is an agile way to manage a project

“The... ‘relay race’ approach to product development...may conflict with the goals of maximum speed and flexibility. Instead a holistic or ‘rugby’ approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today’s competitive requirements.”

Hirotaka Takeuchi and Ikujiro Nonaka, “The New New Product Development Game”, *Harvard Business Review*, January 1986.



Lecture 3 – Intended Learning Objectives

Module 8 - Software Development Lifecycles

Agile:

1. Understand what Agile is and its origins.
2. Understand the Agile framework.
3. Understand Scrum – Roles, Ceremonies and Artefacts.
4. Understand advantages / disadvantages of Agile.

Module 8.3 – Scrum

Scrum in 100 words

- Scrum is an agile process that allows us to focus on delivering the highest business value in the shortest time.
- It allows us to rapidly and repeatedly inspect actual working software (every two to four weeks).
- The business sets the priorities. Teams self-organise to determine the best way to deliver the highest priority features.
- Every two to four weeks, you can see real working software and decide to release it as is or continue to enhance it for another sprint.



Module 8.3 – Scrum is used by many organisations

- Microsoft
- Yahoo
- Google
- Electronic Arts
- High Moon Studios
- Lockheed Martin
- Philips
- Siemens
- Nokia
- Capital One
- BBC
- Intuit
- Intuit
- Nielsen Media
- First American Real Estate
- BMC Software
- Ipswitch
- John Deere
- Lexis Nexis
- Sabre
- Salesforce.com
- Time Warner
- Turner Broadcasting
- Oce

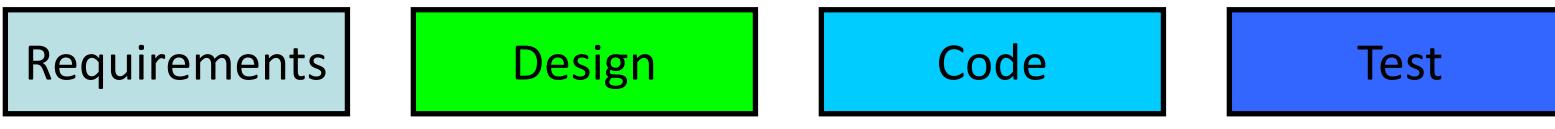
Module 8.3 – Scrum is used for all types of projects

- Commercial software
- In-house development
- Contract development
- Fixed-price projects
- Financial applications
- ISO 9001-certified applications
- Embedded systems
- 24x7 systems with 99.999% uptime requirements
- the Joint Strike Fighter
- Video game development
- FDA-approved, life-critical systems
- Satellite-control software
- Websites
- Handheld software
- Mobile phones
- Network switching applications
- ISV applications
- Some of the largest applications in use

Module 8.3 – Scrum Key Characteristics

- Self-organising teams
- Product progresses in a series of focused sprints
- Requirements are captured as items in a list of product backlog
- Scrum is one of the agile processes – the one most widely used, discussed and debated
- Time frame is contained to a manageable size (weeks or months)

Module 8.3 – Scrum Framework - *Sprints*



Rather than doing one thing at a time...

...Scrum teams do a little of everything all the time

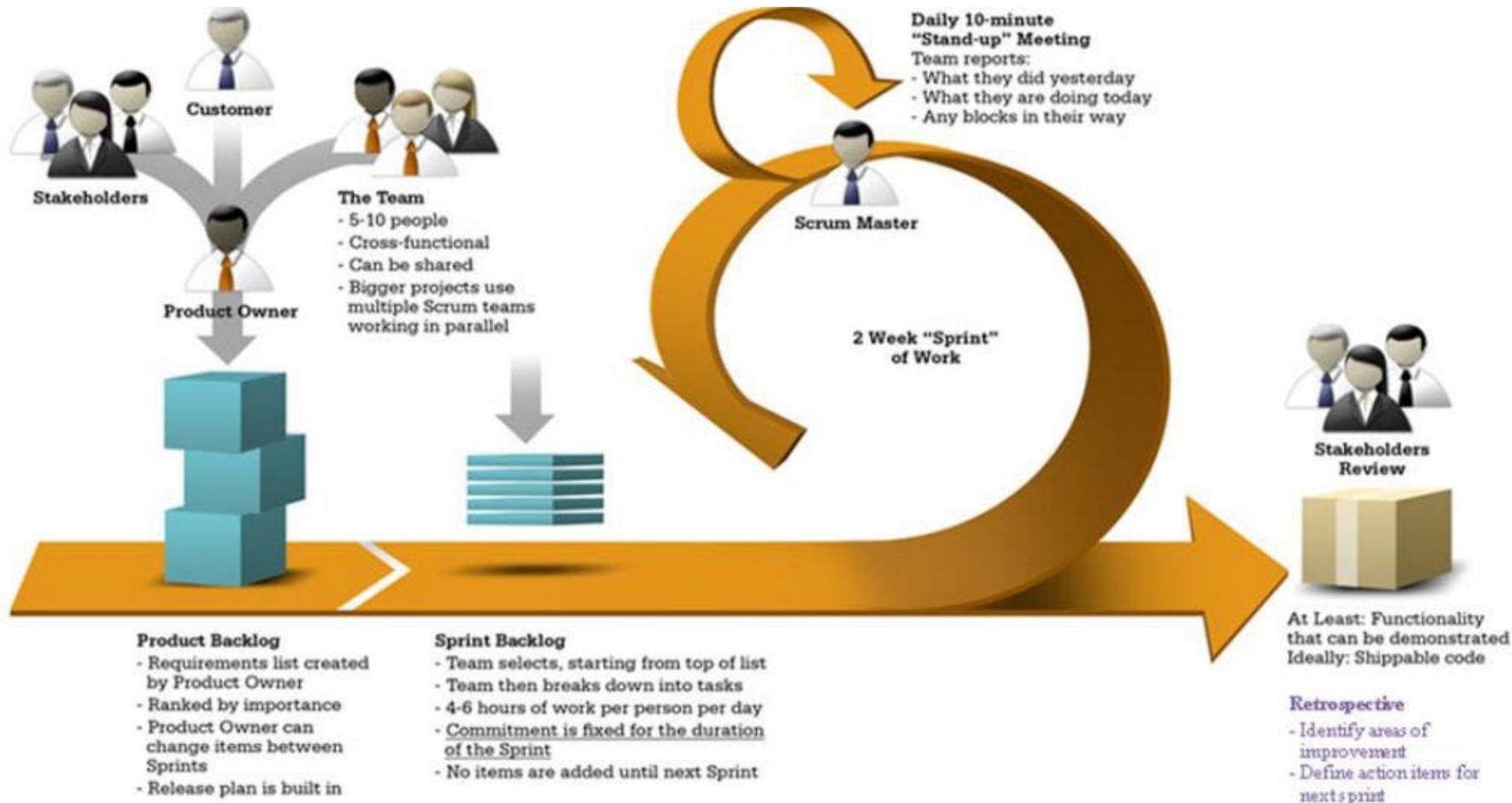


Source: "The New New Product Development Game" by Takeuchi and Nonaka. Harvard Business Review, January 1986.



WILLING TO LEARN

Module 8.3 – Scrum Framework





Module 8.3 – Scrum Framework

Roles

- Product owner
- ScrumMaster
- Team

www.mountaingoatsoftware.com

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily stand-ups

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts



Module 8.3 – Scrum Roles

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

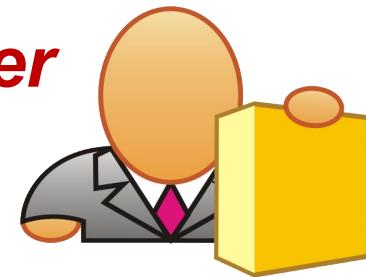
- Sprint planning
- Sprint review
- Sprint retrospective
- Daily stand-ups

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

Module 8.3 – Scrum Roles – *Product Owner*

- Defines the features of the product
- Decides on release date and content
- Is responsible for the Benefits / Profitability of the product (ROI)
- Prioritises features according to market value
- Adjusts features and priority every iteration, as needed
- Accepts or reject work results



Module 8.3 – Scrum Roles – *Scrum Master*



- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments / road blocks
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles
- Shields the team from external interferences
- Is a member & active participant of the Scrum Team

Module 8.3 – Scrum Roles – *The Team*



- Typically 6 - 9 people
- Cross-functional:
 - Programmers, testers, user experience designers, business representatives etc.
- Members should be full-time – some exceptions
- Co-located (physically or virtually)

Module 8.3 – Scrum Ceremonies / Meetings

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily stand-ups

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

Module 8.3 – Scrum Ceremonies / Meetings

Sprint Planning

- Defines how to achieve sprint goal (design)
- Create sprint backlog (User Stories) from product backlog
- Estimate sprint backlog in team velocity and Story Points
- Product Owner priority guides the work
- Release Plan is created
- High-level design is considered

Module 8.3 – Scrum Ceremonies / Meetings

Sprint Planning Meeting

1st Half of the meeting

- Team defines **what** can be done in this sprint
- Starts by writing down the Sprint goal
- Identify the items from the backlog that can achieve this

2nd Half of the meeting

- Team figures out **how** the work will get done
- Break down each item in the sprint backlog into tasks to capture all the work required
- The tasks form the basis of the sprint plan that is used to track, cost and manage progress

Source: Head First Agile – A Brain-Friendly Guide to Agile Principles, Ideas, and Real-World Practices By [Andrew Stellman, Jennifer Greene](#)

Module 8.3 – Scrum Ceremonies / Meetings

Daily Stand-up

- Parameters
 - Daily
 - 15-minutes and no more than 30 mins
 - Stand-up
- Not for problem solving / Not a status meeting
 - Whole world is invited
 - Only team members, ScrumMaster, product owner, can talk
- Helps avoid other unnecessary meetings
- 3 key questions asked:
 1. What did I do yesterday.
 2. What will I do today.
 3. What is in my way to get my work completed.



Module 8.3 – Scrum Ceremonies / Meetings

Sprint Reviews - Showcase

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
- 2-hour prep time rule
- No slides
- Whole team participates
- Invite the world



Module 8.3 – Scrum Ceremonies / Meetings

Sprint Retrospective

- Periodically look at what is and isn't working
- Typically 30 minutes
- Done after every sprint
- Whole team participates:
 - ScrumMaster and Team
- Possibly Product Owner, customers and others
- Discuss what to:
 - Start Doing, Stop Doing and Continue Doing



Module 8.3 – Scrum Artefacts

Roles

- Product owner
- ScrumMaster
- Team

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily stand-ups

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

Module 8.3 – Scrum Artefacts – Product Backlog

User Stories

- A User Story is a requirement expressed from the perspective of an end-user / customer of the system
- User stories shift the focus from writing about requirements to talking about them
- User stories are short, simple descriptions of a feature told from the perspective of the customer who wants the new capability of the system. They follow a simple template:
 - *As a < type of user >, I want < some goal > so that < some reason >*

Module 8.3 – Scrum Artefacts – Product Backlog

User Stories

- User stories are written at varying levels of detail.
- They can cover a large amounts of functionality such as this example from a Professional Membership Website:
 - *As a site visitor, I want to get all information associated with my professional membership so that I have access to all information centrally.*
- Because this level of detail is too large for an agile team to complete in one iteration, it is sometimes split into smaller user stories before it is worked on

Module 8.3 – Scrum Artefacts – Product Backlog

Story Points

- Story points are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a product backlog item or any other piece of work
- Story points help estimate how much work can be done in a sprint
- When estimating with story points, a value is assigned to each item. The raw values are unimportant, what matters are the relative values
- A story that is assigned a 2 should be twice as much as a story that is assigned a 1. It should also be two-thirds that is estimated as a 3 story point.
- Instead of assigning 1, 2 and 3, that team could assign 100, 200 and 300. Or 1 million, 2 million and 3 million. It is the ratios that matter, not the actual numbers



Module 8.3 – Scrum Artefacts – Product Backlog



Product Backlog

- User Story 1
- User Story 2
- User Story 3
- User Story 4
- User Story 5
- User Story nn

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Product Backlog User Stories are selected for a Sprint by Product Owner
- Reprioritised at the start of each sprint



Module 8.3 – Scrum Artefacts – Product Backlog Example - Professional Body Website

Product Backlog

User Story 1

User Story 2

User Story 3

User Story 4

User Story 5

User Story 6

News Section – Sprint 1

- User Story 1 - As a site visitor, I can read current news on the home page so that I stay current on key professional items
- User Story 2 - As a site visitor, I can email news items to the editor so that they can be considered for publication
- User Story 3 - As a site member, I can subscribe to an RSS feed of news so that I can stay current on the news that is of interest to me

Courses and Events – Sprint 2

- User Story 4 - As a site visitor, I can see a sorted list by date of all upcoming “Certification Courses” so that I can choose the best course for me
- User Story 5 - As a site visitor, I can see a list of all upcoming “Other Courses” (non-certification courses) so that I can choose the best course for me
- User Story 6 - As a site visitor, I can see a list of all upcoming “Social/Networking Events.” so that I can select ones I am able to attend

www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog/example

Module 8.3 – Scrum Artefacts – Product Backlog Example - Professional Body Website

Product Backlog



News Section - Total of 6 Story Points to complete Sprint 1

- User Story 1 - As a site visitor, I can read current news on the home page so that I stay current on key professional items – 1 Story Points
- User Story 2 - As a site visitor, I can email news items to the editor so that they can be considered for publication – 2 Story Points
- User Story 3 - As a site member, I can subscribe to an RSS feed of news so that I can stay current on the news that is of interest to me – 3 Story Points

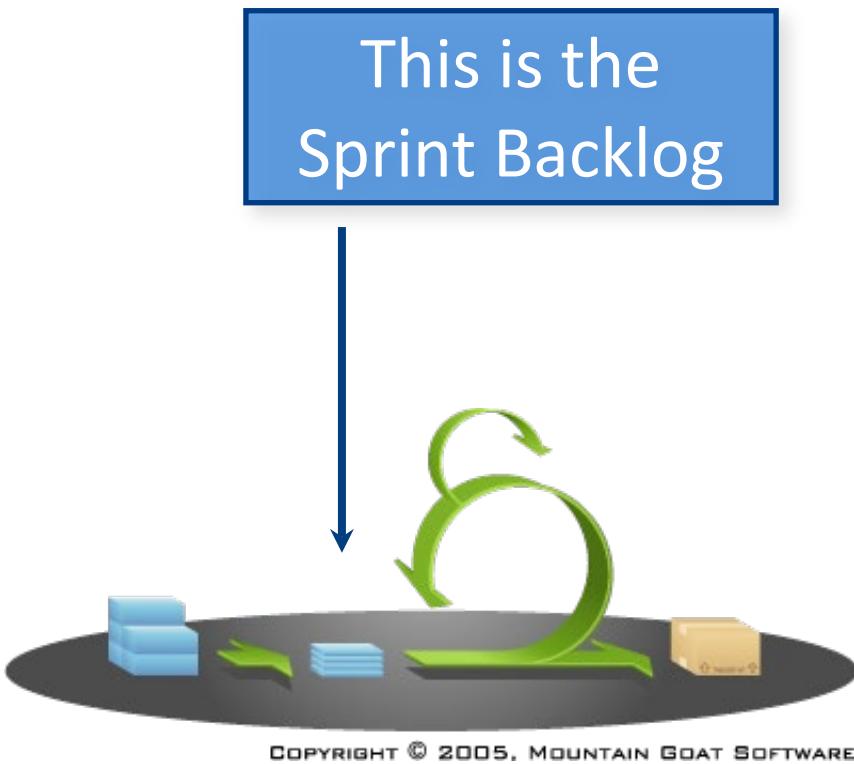
Courses and Events - Total of 8 Story Points to complete Sprint 2

- User Story 4 - As a site visitor, I can see a sorted list by date of all upcoming “Certification Courses” so that I can choose the best course for me – 2 Story Points
- User Story 5 - As a site visitor, I can see a list of all upcoming “Other Courses” (non-certification courses) so that I can choose the best course for me – 5 Story Points
- User Story 6 - As a site visitor, I can see a list of all upcoming “Social/Networking Events.” so that I can select ones I am able to attend – 1 Story Points

To complete this total Product Backlog would take 14 Story Points

www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog/example

Module 8.3 – Scrum Artefacts – Sprint Backlog / User Story

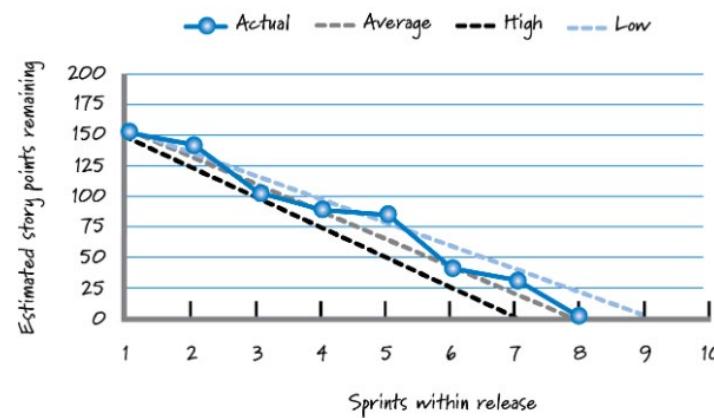


- Scrum team decompose User Stories to Low level User Stories during Sprint Planning
- The User Stories are used for a conversation between the SME and developer. Developer updates the User Stories with the tasks and hours estimates, "Just-In-Time"
- Remaining estimated items are updated daily
- Sprint Backlog is seldom altered
- User stories in the sprint are either completed 100% or not done



Module 8.3 – Scrum Artefacts – Burn Down Chart

- A burn down chart is a graphical representation of work left to do versus time.
- The outstanding work (or backlog of user stories) is often on the vertical axis, with time along the horizontal.
- It is used to predict when all of the work will be completed.



Module 8.3 – Scrum Summary

- Another look at **SCRUM**

<https://www.youtube.com/watch?v=9TycLR0TqFA>

Lecture 3 – Intended Learning Objectives

Module 8 - Software Development Lifecycles

Agile:

1. Understand what Agile is and its origins.
2. Understand the Agile framework.
3. Understand Scrum – Roles, Ceremonies and Artefacts.
4. Understand advantages / disadvantages of Agile.

Module 8.4 – Agile – Advantages & Disadvantages

Advantages

- Customer satisfaction by rapid, continuous delivery of usable software
- People and interactions are emphasised rather than process and tools
- Continuous attention to technical excellence, good design and quality
- Regular adaptation to changing circumstances

Disadvantages

- Difficult to assess the effort required at the beginning
- Can be very demanding (from traditional approaches) on users time
- Harder for new starters to integrate into the team
- Agile is a very different approach – It can be intense for the team
- Requires experienced resources (which are limited in today's market)

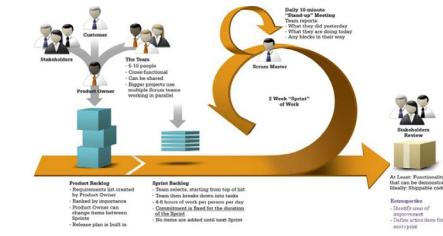
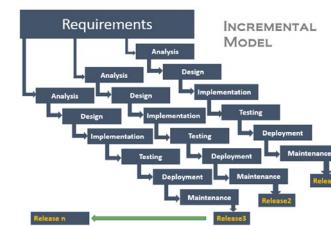
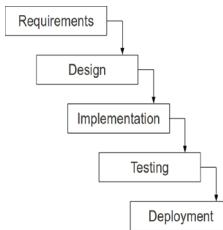
WILLING TO FAIL

Module 8.4 – Formal or Agile which one Should I use???



There is no one right answer. The following questions can assist deciding:

- How Stable Are the Requirements?
- Do the end users need to collaborate?
- Is the Time Line Aggressive or Conservative
- What Is the Size of the Project
- Where Are the Project Teams Located
- What Are the Critical Resources?



WILLING TO LEARN

Agile Where to find out more information

- www.agilealliance.org
- www.mountaingoatsoftware.com/scrum
- www.scrumalliance.org
- www.controlchaos.com



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 3

Lecture 3 – Intended Learning Objectives

**Module 7 – Software Development
Lifecycles - Formal.**

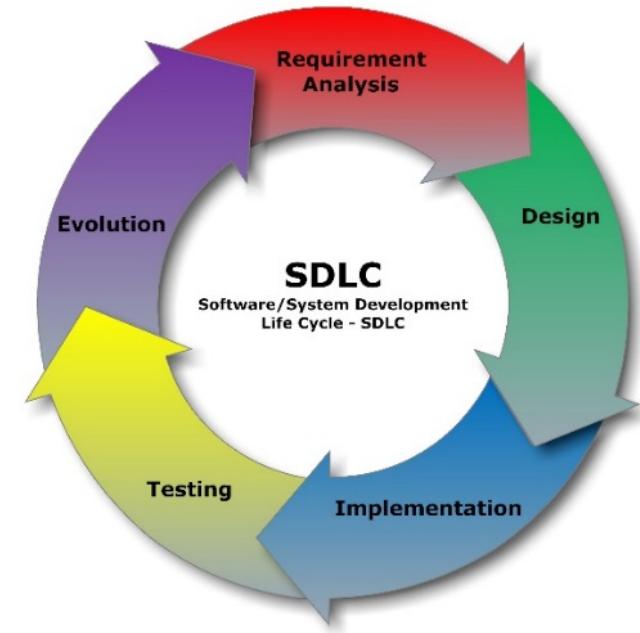
**Module 8 – Software Development
Lifecycles - Agile.**

Module 7.1 – Software Development Life Cycle (SDLC)

The systems development life cycle (SDLC), also referred to as the application development life-cycle, is a term used in systems engineering, information systems and software engineering to describe a **process** for planning, creating, testing and deploying an information system.

Activities in SDLC:

- Requirements gathering
- Systems / Architectural Design
- Implementation / coding / Integration
- Testing
- Evolution:
 - Delivery and Release - Deployment
 - Maintenance





Module 7.1 – SDLCs

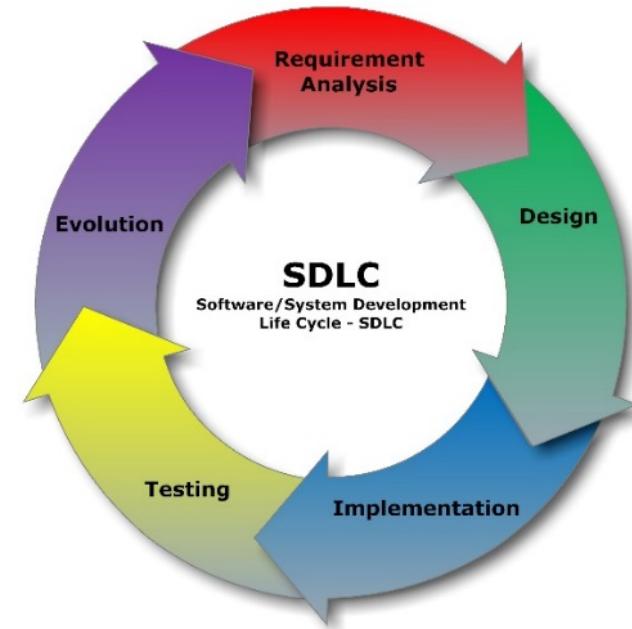
There are many SDLCs around with organisations typically favouring a blend of Formal and Agile approaches.

1. Formal

- Waterfall
- Incremental
- V-Model

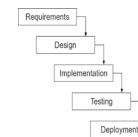
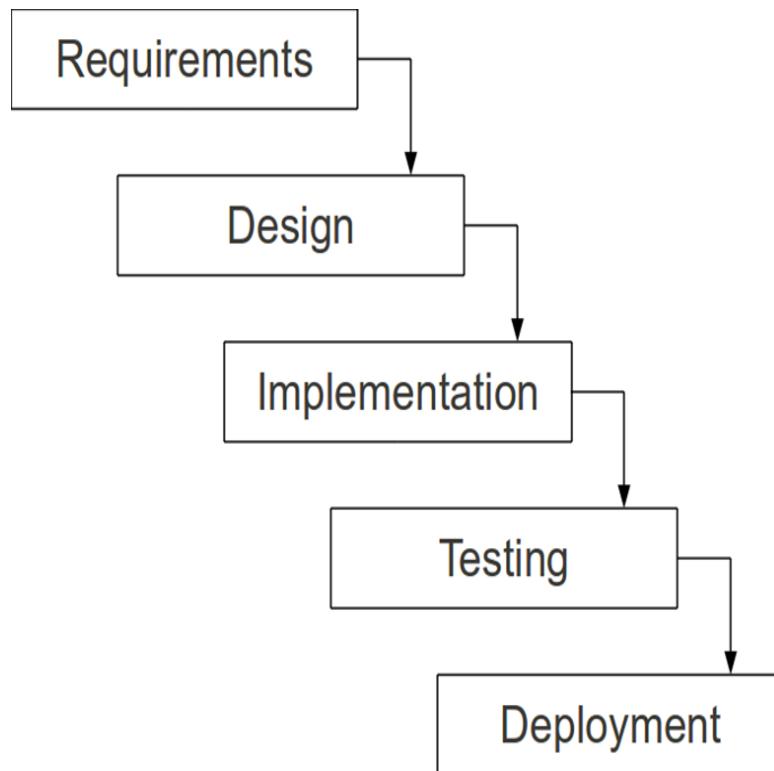
2. Agile

- Scrum
- Kanban
- Extreme Programming





Module 7.1 – Waterfall



Advantages

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model
- Phases are processed and completed one at a time
- Documentation available at the end of each phase
- Works well for projects where requirements are very well understood and remain stable

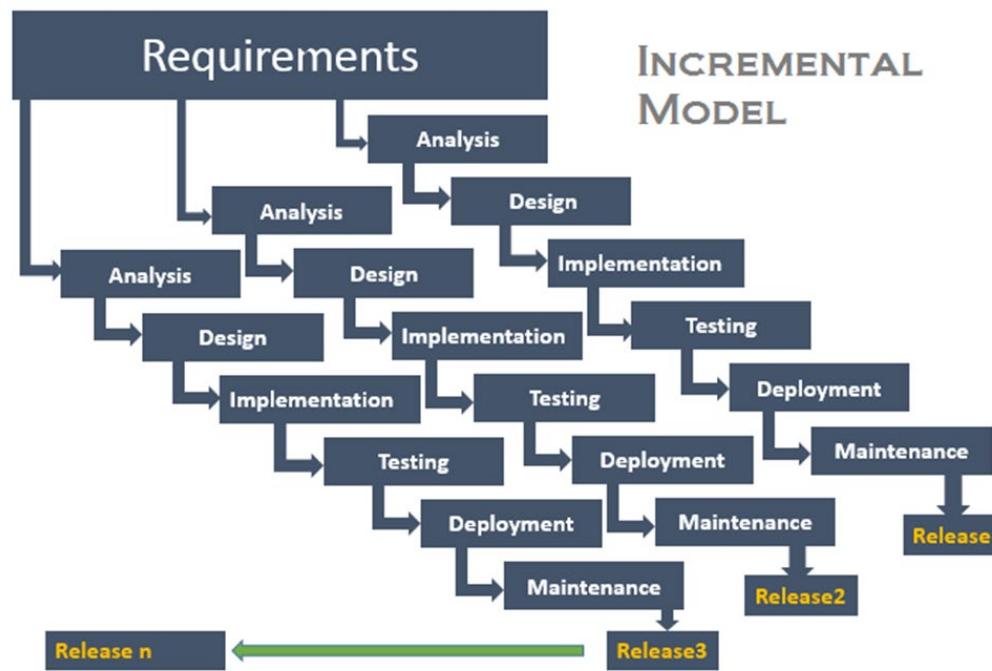
Disadvantages

- Difficult to accommodate change after the process is underway
- One phase must be completed before moving on to the next
- Unclear requirements lead to confusion
- Client approval is in the final stage
- Difficult to integrate risk management due to uncertainty

Module 7.1 – Incremental Model



In incremental model the **whole requirement** is divided into various releases. Multiple cycles take place, making the life cycle a **multi-waterfall** cycle. Cycles are divided up into smaller, more easily managed modules.





WILLING TO LEARN

Module 7.1 – Incremental Model



Advantages – compared to standard waterfall

- Each release delivers an operational product
- Less costly to change the scope/requirements
- Customers can respond to each build
- Initial product delivery is faster
- Customers get important functionality early
- Easier to test and debug during smaller iterations

Disadvantages – compared to standard waterfall

- More resources may be required
- More management attention is required
- Defining / partitioning the increments is difficult and often not clear
- Each phase of an iteration is rigid with no overlaps
- Problems may occur at the time of final integration

Module 7.1 – Formal Models



Characteristics where “Formal” Models make sense:

- Projects where the customer has a very clear view of what they want
- Projects that will require little or no change to requirements
- Software requirements are clearly defined and documented
- Software development technologies and tools are well-known
- Large scale applications and systems developments



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 6



WILLING TO FIGHT

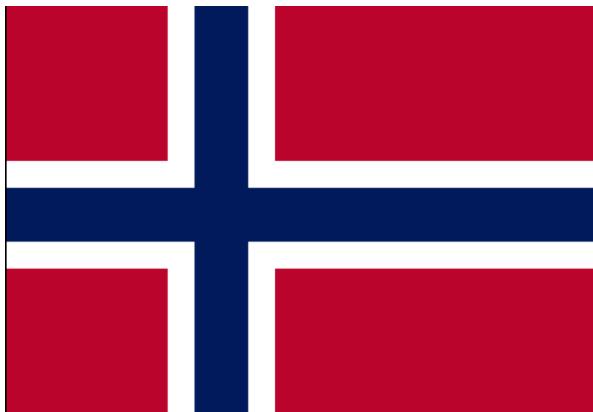
Positive or negative amounts

() = negative

No brackets = positive



WILLING TO FIGHT



https://en.wikipedia.org/wiki/Norway#/media/File:Flag_of_Norway.svg

<https://www.fjordnorway.com/top-attractions/vikings>

Case Study

SWEN90016

	Organisation A (150 employees in four organizational units)		Organisation B	
	Project A	Project B	Project C	Project D
Project size	3 years	3 years	9 months	12 months
Developers	7 (Norway: 4, India: 3)	5 (including 2 consultants)	6	4
Scrum master	One of the developers	One of the developers	From the quality and marketing department	The department head
Product owner	From the sales and business department	Former project manager	Former project manager, in another city	Former project manager, in another city
System developed	Information system for integrity management of pipelines, both off-shore and on-shore	Information system for designing and maintaining off-shore installations	Geographical information system for planning and coordination work	Information system for handling reports from clients

[1]

Case Study

SWEN90016

	Organisation A (150 employees in four organizational units)		Organisation B	
	Project A	Project B	Project C	Project D
Scrum introduced	Middle of project	Middle of project	Beginning of project	Beginning of project
Scrum practices	All. No retrospective in every sprint	All	All. No retrospective in every sprint	All. No retrospective in every sprint
Physical wall with tasks	No	Yes	No	Yes

[1]



[1]

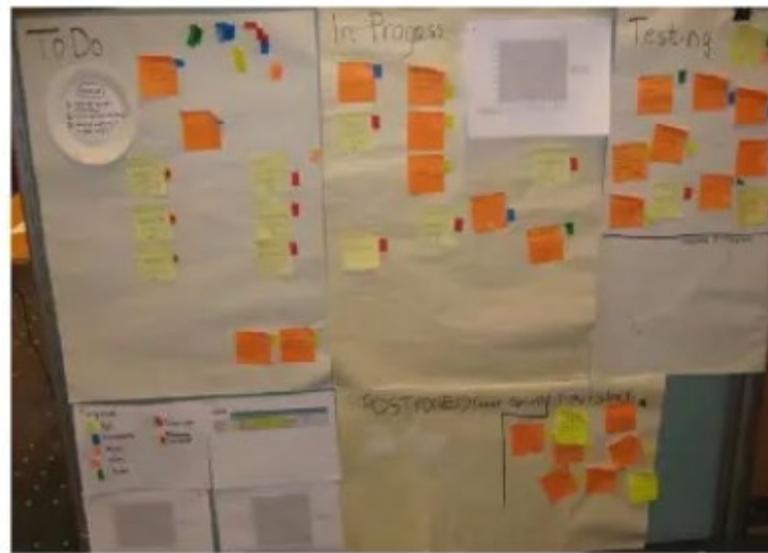


Fig. 1. The Scrum wall of project B showing the task status and task flow.



Product backlog and scope creep

The Scrum master said:

- “Everything cannot be equally important. The product-backlog makes it easier for us to tell the Product owner to prioritize what is most important. The prioritized backlog helped in aligning all input to the project”

A developer said:

- “The sprint and sprint backlog makes it easier to say “no” to the Product owner. . . ”

(Project A)

[1]



Resourcing

Scrum master

- “It is now more difficult to “steal” resources from us, because the consequences of losing resources during a sprint are more visible with Scrum. Earlier the deadline was 6–12 months ahead, and it was easy to steal a day or two. There has been a change of attitude in the company, and it is now well accepted that you do not steal resources from a Scrum team during a sprint”

(Project A)

[1]



Resourcing

- Lost resources during iterations because of conflicting priorities within the company. The teams were lacking support from the organization and adequate resources, which are two important factors for achieving self-management. Not aligning tactical decisions on the company level resulted in challenges on the project and product level.

(Project C and D)

[1]



Planning

A developer said:

- When we use planning-poker we do it too fast and without preparation , we only suggest numbers without really knowing. The tasks are more complex than we realize. Because we lack knowledge about the problem to solve, the meetings are time consuming. And when I propose an estimate I do not get really valuable feedback from the others.
- When it comes to the daily Scrum, I do not pay attention when Annis talking. For me it is a bit far off what she talks about, and I do not manage to pay attention. She talks about the things she is working on. I guess this situation is not good for the project.

(Project B)

[1]



Sprints

Third, projects C and D were missing a clear definition of ‘done’ and the Scrum master often wanted to make the team look better than they actually were. Therefore, each iteration started by performing tasks that were officially finished in the previous iteration, and then everybody in the team knew they could not complete what was planned for the current iteration

A developer said:

- Since we also added so many features we knew we could not finish, we did not care if we did not complete all the tasks during a sprint. Some tasks were moved four sprints before they were even started

(Project C and D)

[1]



Sprints

The team moved from Waterfall to Agile

The scrum master said

- I perceive the team as not taking real responsibility. When they have finished a task they ask me “what should I do now?”. I would expect them to take more responsibility. They are used to be given single tasks. This is how we used to work. They do not see all the tasks as a pool they can choose from

(Project B)

[1]

[1] Nils Brede Moe, Aybüke Aurum and Tore Dybå , “Challenges of shared decision-making: A multiple case study of agile software development,” *Information and Software Technology*, vol. 54, (8) pp. 853-865, 2012.



SWEN90016

Software Processes & Project Management

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne mzalk@unimelb.edu.au

Copyright University of Melbourne 2020

2021 – Semester 1
Lecture 6

Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. Individuals & motivation.
2. Organisational theory and motivation.
3. Project Management & Leadership.
4. Teams why we use them and their value.
5. Teams forming and performing.
6. Team structures.
7. Advantages & disadvantages of teams.



Module 9.1 – Individuals and Motivation

So why is this important to Project Management?

L1.3 – What is Project Management

Project Management is the planning, delegating, monitoring and controlling of all aspects of the project, and motivating those involved to achieve the project objectives within the expected targets for time, costs, quality, scope, benefits and risks.

Value lies in:

- Organising and structuring scarce resources
- Managing risk
- Identifying and clearing issues
- Managing and implementing change
- Retaining and re-using knowledge
- Organisational wide learning from past success and failures

L1.5b – Project Manager Key Activities “*a change is occurring*”

Agile is redefining the way we execute projects and the role of the PM. In Agile:

- No defined PM role
- Key activities are spread / shared across team members
 - Key project activities are still undertaken formally with appropriate documentation
- Some alignment to activities of a Scrum Master
- Move from Command and Control to Servant Leadership
 - Coaches and facilitates teams to deliver
 - Emphasises objectives
 - Is invested in the program's overall performance
 - Asks the teams for answers
 - Allows the teams to self-organise and hit their stride
 - Assists others with fixing issues

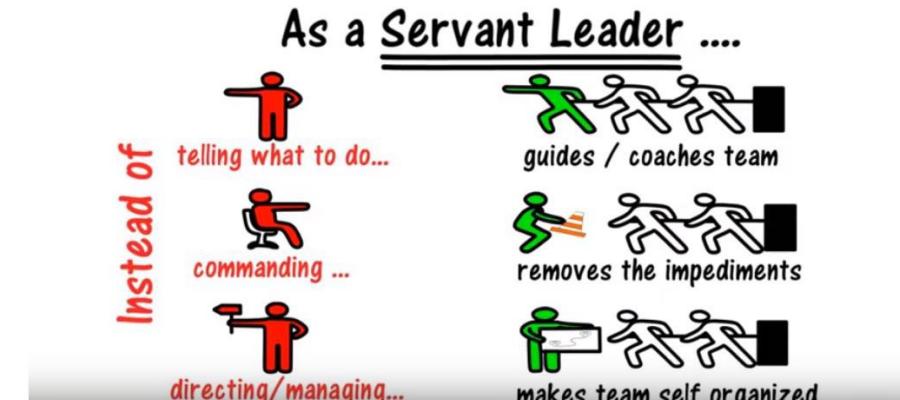


Module 9.1 – Individuals and Motivation

.....and even more critical in an agile world!

A "Scrum Master" represents a pattern known as **Servant Leadership**.

They manage teams not by telling them what to do, but by removing impediments that get in their way and by coaching them in best practices.



www.scrum.org/resources/blog/scrum-master-servant-leader

Module 9.1 – Individuals and Motivation

- Motivation *is derived from the word ‘motive’*
- It is the driving force within individuals that propels them into action

Motivation can inspire, encourage, and stimulate individuals and project teams to achieve great accomplishments. Motivation can also create an environment that fosters teamwork and collective initiatives to reach common goals or objectives.

www.pmi.org/learning/.../motivation-increase-project-team-performance-7234

Module 9.1 – Individuals and Motivation

- Motivation *is derived from the word ‘motive’*
- It is the driving force within individuals that propels them into action

What motivates you?



- Motivation *is derived from the word ‘motive’*
- It is the driving force within individuals that propels them into action

What motivates you?





- Motivation *is derived from the word ‘motive’*
- It is the driving force within individuals that propels them into action

What motivates you?



What motivates you?



Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. **Organisational theory and motivation.**
3. Project Management & Leadership.
4. Teams why we use them and their value.
5. Teams forming and performing.
6. Team structures.
7. Advantages & disadvantages of teams.

Module 9.2 – Organisational Theory & Motivation

Organisational theory consists of approaches to organisational analysis. Organisations are groups of individuals that are structured and managed to meet a need, or to pursue collective goals.

Some well used approaches:

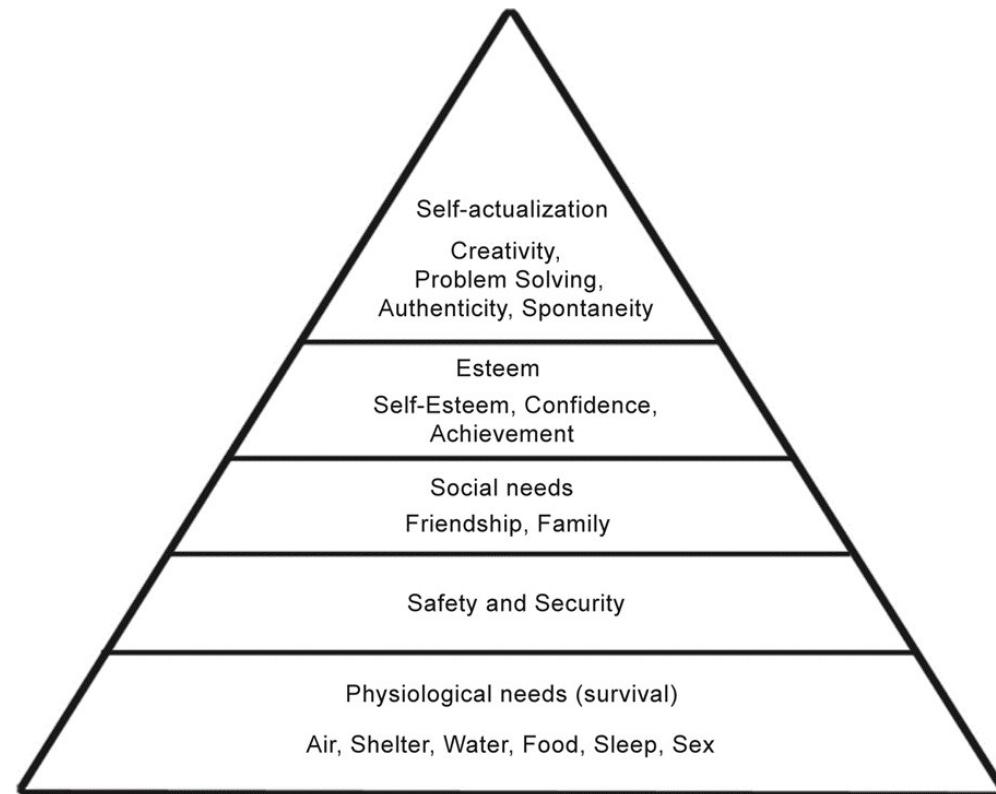
- Maslow Hierarchy of Needs
- Herzberg



Module 9.2 – Organisational Theory & Motivation

Maslow Hierarchy of Needs:

- Proposed by Abraham Maslow in 1943 and still widely used today
- A key tool used by managers in how individuals are motivated
- Focuses on a 5 tier model of human needs
- Describes humans are motivated to achieve certain needs
- Needs take precedence over others and the basic needs must be more or less met before higher needs
- Individual behaviour is multi-motivated and stimulated by more than one need





Module 9.2 – Organisational Theory & Motivation

Hertzberg Two Factor Theory:

- Proposed by Fredrick Hertzberg in 1959 and still widely used today
- Asked people to describe situations when they felt really good and really bad about their jobs
- There are a set of factors in the workplace that cause satisfaction
- And a separate set of factors that cause dissatisfaction
- Remedyng the causes of dissatisfaction will not create satisfaction





Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. **Project Management & Leadership.**
4. Teams why we use them and their value.
5. Teams forming and performing.
6. Team structures.
7. Advantages & disadvantages of teams.



Module 9.3 – Project Management & Leadership



Project Management & Leadership - motivates and demotivates the team!

- Project Managers must Manage & Lead
- Management is the process where resources are used and decisions made in order to achieve the goal
- Managers set objectives and decide how to achieve them
- Leadership is the ability to influence and direct people to achieve a common goal
- Leaders inspire and motivate people to meet goals

Team Member	Role	Location	Development Goal	Motivation Drivers – Employee wants/needs	How to apply motivational driver
Marion	Technical Expert	Melbourne	Project Management	Formal training and recognition	Sponsor Marion in Melbourne Uni Masters program
Harry	Developer	Gold Coast	Detailed Technical knowledge	Work Life Balance	Allow 4 day per week work in line with Project activity

Module 9.3 – Project Management Common Motivational Mistakes

- Whatever motivates me will motivate others
- People are motivated primarily by money
- Team members love to receive formal awards
- Give them a rally slogan
- The best project leader is a strong cheerleader
- These people are professionals. They don't need motivating
- I'll motivate them when there is a problem
- I'll treat everyone the same. People like that, and it will motivate them

Reference: Flannes, S. W. & Levin, G. (2005). *Essential People Skills for Project Managers*. Vienna, VA: Management Concepts, Inc.

Module 9.3 – Project Management & Leadership Summary

- Individuals are individuals and we are all motivated by different means
- Projects succeed / fail because of people so manage, lead and motivate them to increase success
- Leadership and Management are different. Consciously select the style that is right for the situation
- The biggest impact you can have is by managing yourself take the necessary step to achieve this

Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. ~~Project Management & Leadership.~~
4. **Teams why we use them and their value.**
5. Teams forming and performing.
6. Team structures.
7. Advantages & disadvantages of teams.

Moule 9.4 - Teams and Teamwork

We have all experienced this!



Module 9.4 - Teams



Teamwork in the workplace is an **critical** factor for project **success**. As a result, developing an **effective** project team is one of the **primary responsibilities** of a project manager. Teamwork creates human synergy.

I prefer to work.....

By myself /
Alone

As part of a
diverse
team



- An individual is a person with a unique set of skills
- A Group is a collection of people working together who do not necessarily work collectively toward the same goal
- A Team is two or more individuals consciously working together to achieve a common objective
- A Group becomes a Team when members demonstrate a commitment to each other and to the end goal toward which they are working



Module 9.4 – Why Teams



1. **Very few (if any) individuals** possess all the knowledge, skills, and abilities needed to accomplish all tasks.
2. **Complementary teamwork skills** are one of the most commonly **required** skills in the work environment.
3. Substantial **benefits** to the organisation and to the team members.
4. **Shared accountability** increases likelihood of **success**.



Module 9.4 – Why Teams - Benefits

1. **Enhanced Opportunities:** Individuals & organisation.
2. **Greater Productivity:** Leverage the strengths and skills of the collective group.
3. **Increased Ownership & Accountability:** Multiple people collectively owning the activity and the outcome.
4. **More Creativity and Innovation:** Individuals build upon one another's ideas with solutions going beyond one person's vision of what's possible.
5. **Greater Joy and Satisfaction Among Team Members:** A space for people to socialise, connect and be part of something bigger.
6. **Broader Perspective:** Ability to leverage the collective perspective of all team members.
7. **Increased Representation:** Involvement of multiple stakeholders groups and their input.
8. **Increased Equality:** Individuals across all levels can more freely offer their ideas, knowledge and concerns.
9. **More Dialogue:** Teams offer a site where people can voice their feelings, disagreements, opinions and ideas.

Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

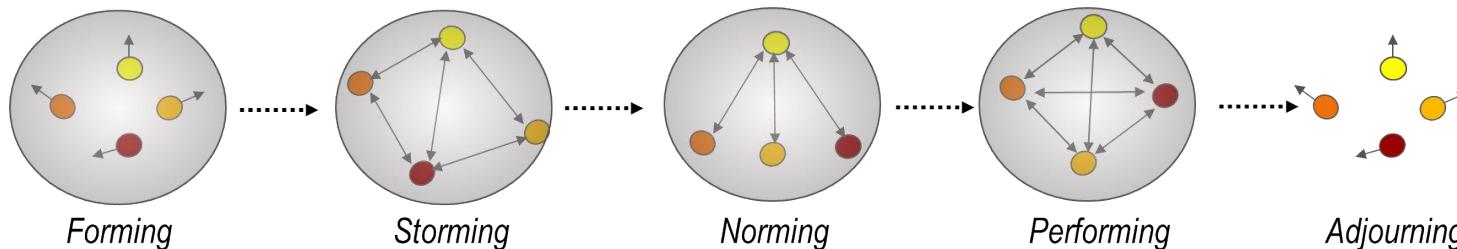
1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. ~~Project Management & Leadership.~~
4. ~~Teams why we use them and their value.~~
5. **Teams forming and performing.**
6. Team structures.
7. Advantages & disadvantages of teams.



Tuckman's Team Development Model



- First proposed by Bruce Tuckman in 1965
- Elegant and simple explanation of team development
- Initial model focused on 4 stages Forming–Storming–Norming- Performing
- Tuckman stated that all phases are necessary and inevitable if a team is to grow, face up to challenges, tackle problems, find solutions, plan work and deliver results
- He found that it was critical for team to go through an Adjourning stage which was added in the 70's





Tuckman's Team Development Model

- As teams develop maturity and ability – leadership styles change and behaviors change
- Tuckman also found that Team Effectiveness changed over time with the team experiencing initial decline in performance after Stage 1 – Forming



<https://project-management.com/the-five-stages-of-project-team-development/>



Module 9.5 – How teams Form & Perform



FORMING

Establishing ground rules and preserving formalities



STORMING

Members communicate, but maintain strict individuality



NORMING

Team bonding and higher acceptance of perspectives



PERFORMING

Less emphasis on hierarchy and more on flexibility



ADJOURNING

Yearly assessment and plan for acknowledging individual contributions



High dependency on the leader

Leader coaches and support

Leader moves to facilitator and enabler

Leader delegates and oversees

Leader acknowledges, recognises and directs



Positive signs

- Clear communication
- Regular brainstorming with all members participating
- Consensus among team members
- Problem solving done by the group
- Commitment to the project outcomes and the other team members
- Regular team meetings are effective and inclusive
- Timely hand off from team members to others or early advise if this won't happen
- Positive, supportive working relationships

Not so Positive signs

- Lack of communication
- No clear roles and responsibilities
- Work is “thrown over the wall”, with lack of concern for timelines or work quality
- Team members work alone, rarely sharing information and offering assistance
- Blame for what goes wrong, no one accepts responsibility
- Lack of support for others
- Frequently absent impacting time and creating additional work for others

<https://www.pmi.org/learning/library/team-building-development-project-management-5707>

Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. ~~Project Management & Leadership.~~
4. ~~Teams why we use them and their value.~~
5. ~~Teams forming and performing.~~
6. **Team structures.**
7. Advantages & disadvantages of teams.

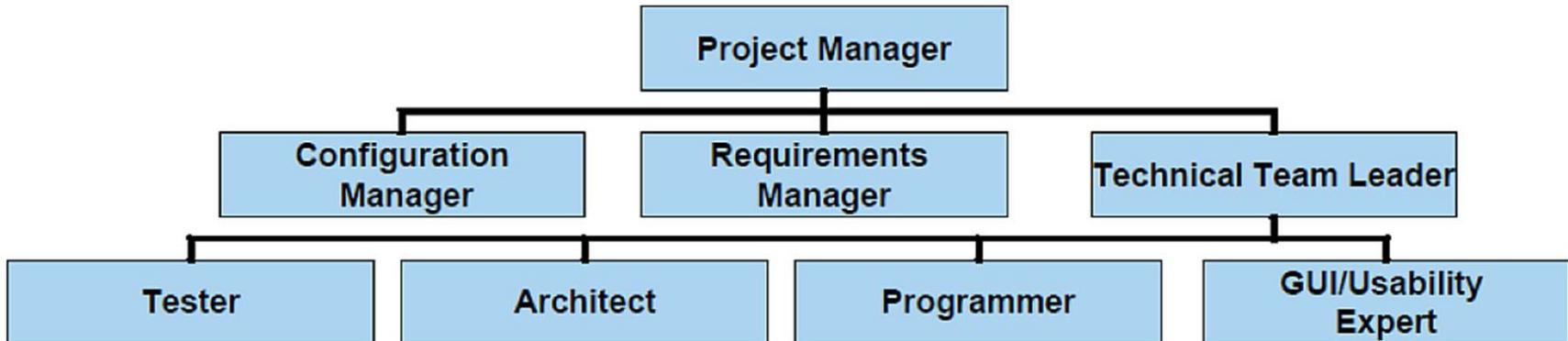


Module 9.6 – Team Structures



Controlled Centralised

- Leader coordinates tasks and directs work
- Communication and Control are vertical
- Sub-teams with leaders to direct and guide sub-groups



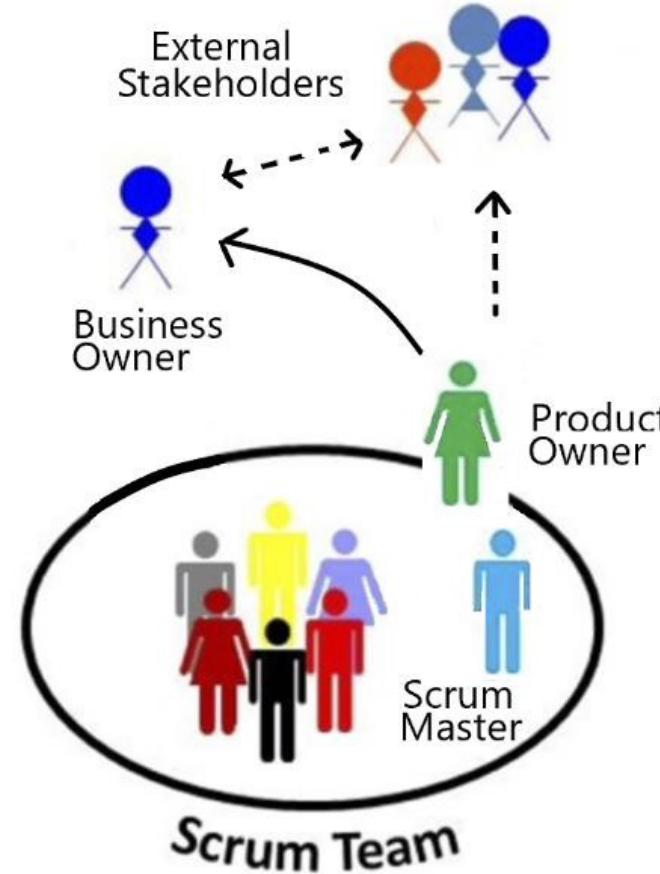


Module 9.6 – Team Structures



Scrum Team

- Used in Agile





Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. ~~Project Management & Leadership.~~
4. ~~Teams why we use them and their value.~~
5. ~~Teams forming and performing.~~
6. ~~Team structures.~~
7. **Advantages & disadvantages of teams.**



AVAILABILITY

Module 9.7 – Teams Advantages / Disadvantages

Advantages

- Provides a larger pool of ideas – creative & collective problem solving
- Interaction enhances the knowledge of the whole team
- Individuals working together can stimulate performance, motivation and output
- Provides continuity across the tasks if people leave
- Increased ownership of the overall outcome & not just the individual component

Disadvantages

- It takes time, effort and great skill to effectively manage
- Some individuals find it difficult and may become overshadowed / overwhelmed
- Unequal involvement - Some people may not pull their weight
- One person can demoralise the whole team
- Social loafing
- Group think



Intended Learning Objectives

Module 9 – Individuals, Motivation and Teams.

1. ~~Individuals & motivation.~~
2. ~~Organisational theory and motivation.~~
3. ~~Project Management & Leadership.~~
4. ~~Teams why we use them and their value.~~
5. ~~Teams forming and performing.~~
6. ~~Team structures.~~
7. ~~Advantages and Disadvantages of teams~~

SDLCs Case Study

*Marion Zalk
Department of Computing and Information Systems
The University of Melbourne
mzalk@unimelb.edu.au*



Background

Early '90s, sending and receiving payments online between individuals was not an easy process.

People didn't even imagine the possibility of such transactions, even though banks and other financial institutions had the means of conducting them.

Used in 200+ countries in the world.

50+ currencies



MELBOURNE

A success story An example of first mover advantage.

- 1998: Initial product created to encrypt word documents
But no one wanted to keep word documents secret
Encrypted money transfer between PalmPilots
The first tool to transfer money electronically
Built web interface as a sales demo tool **ACCIDENT**
PalmPilot device not popular, so demo-ed the tech on the web
- 1999: Encrypted money transfer **service** component
- 2000: Merge with an online bank (Elon Musk- X.com)
- 2001: Changed name to PayPal (Confinity + X.com)
- 2002 Listed on NASDAQ; Acquired by Ebay



Slow down

2002 -

Company was small, they could rapidly build solutions which put them at the leader

Innovation began to decrease, release cycles increased from weeks to months and many months

2008 - 6 key drivers for the slow down

1. Annual Planning –defined and documented in a Product Requirements Document (PRD) which could take months and were often out of date before coding began.

[1]



Slow down

2. Complex quarterly planning

New process for planning to allow for more flexibility, added complexity

3. Domain bottlenecks

Divided the environment into 85 domains, unintended consequence was a bottleneck at certain key areas (necessary for every project). These included risk, compliance, payments.

4. Developer context switching

To enable more projects to kick-off, developer were assigned to multiple projects during planning. This context switching reduced productivity of the individual and team.

[1]



Slow down

5. Traditional waterfall development

To facilitate and coordinate efforts across the organisation, they used waterfall. Those teams that had developed using grass root agile efforts were forced to follow the strict process of Waterfall.

6. Integration testing cycles

To ensure quality of each release- rigorous testing required, code branches grew and integration and testing cycles increased even for the smallest change- 6 weeks of integration testing cycle to ensure it didn't break the site.



Results

People (developers) were frustrated.

Exec management was frustrated.

Customers were frustrated

What happened?

[1]



Results

2012

PayPal promoted David Marcus- President
Focus was on the customer

Kirsten Wolberg joined as Lead Technology Business Operations
(had experience with agile as CIO at salesforce.com)

Alignment executive sponsorship and leadership with the employees and this enabled PayPal to embark on a radical transformation.

[1]



References

[1] (10 March 2021) PayPal Enterprise Transformation

Available:

https://www.paypalobjects.com/webstatic/en_US/mktg/pages/stories/pdf/paypal_transformation_whitepaper_sept_18_2015.pdf

Risk Management Case Study

*Marion Zalk
Department of Computing and Information Systems
The University of Melbourne
mzalk@unimelb.edu.au*



MELBOURNE



Risk Planning → **Risk Management Plan**

Risk Identification:

Kinds of Risks:

Project
Product
Business

Identification Techniques:

Pondering	Interviewing
Brainstorming	Checklists
Delphi	SWOT Analysis



Risk identification techniques

- Pondering
- Interviewing
- Brainstorming (WBS) learn about in project scheduling
- Checklists:
- Delphi Technique:
- **SWOT Analysis**



Case study

MELBOURNE

Understand the business and IT risk management

1. Small projects
2. Big organisations



- Pondering
 - This simply involves an individual taking the “pencil and paper” approach of risk identification, which involves sitting and thinking about the possible risks that could occur in the project
 - This is one of the initial risk assessment tasks used in many projects
- Interviews/questionnaires
 - Interviewing project stakeholders, or asking them to fill out questionnaires, to harness their knowledge of a domain
 - It is unlikely that a risk manager in a software project will have sufficient knowledge of the methods and tools to be employed to provide a comprehensive view of the risks, so input from stakeholder and domain experts is essential



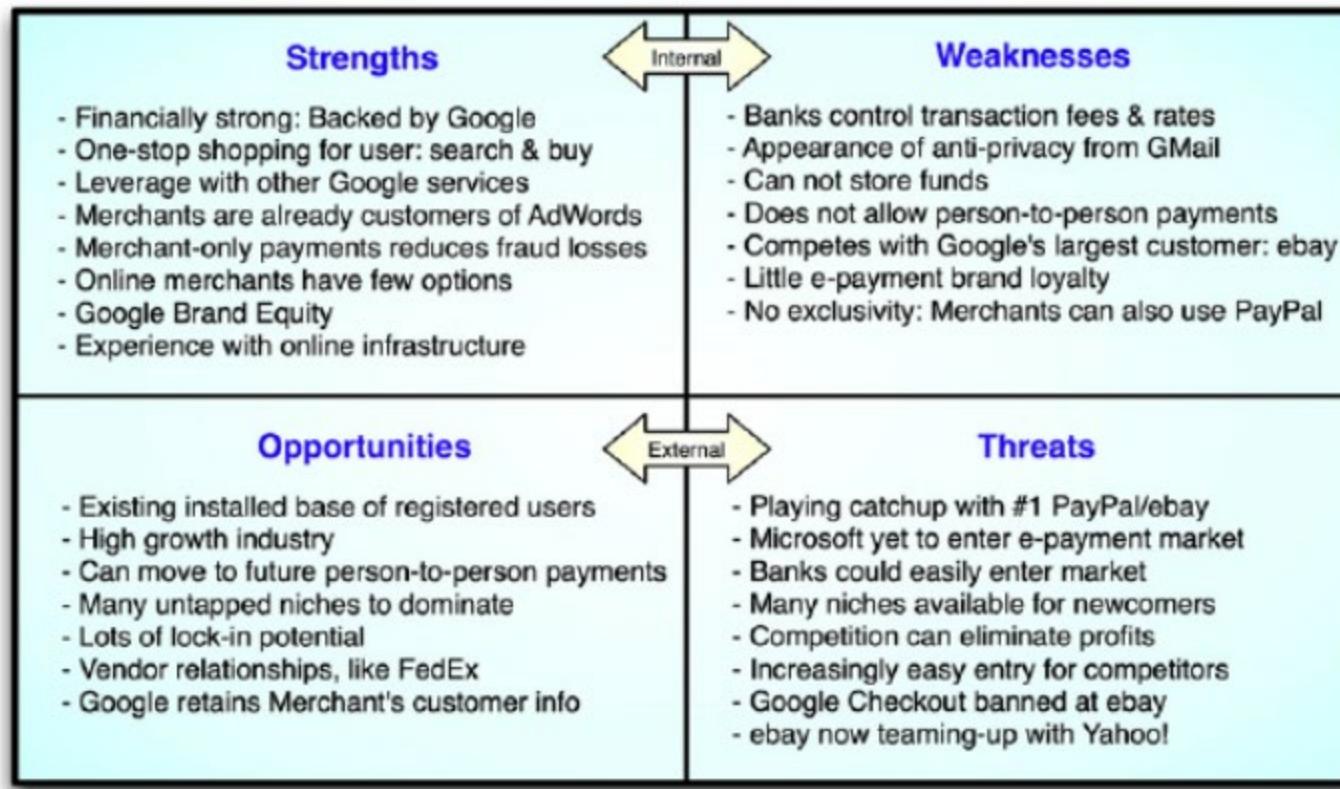
- Brainstorming –
 - The team can use a *risk framework* or the *Work Breakdown Structure (WBS)* to identify threats and opportunities
 - The key is to encourage contributions from everybody
 - The group then discuss and evaluate
- Checklists
 - This involves the use of standard checklists of possible risk drivers that are collated from experience
 - These checklists are used as triggers for experts to think about the possible types of risks in that area



- Delphi Technique
 - A group of experts are asked to identify risks and their impact
 - The responses are then made available to each other anonymously
 - The experts are then asked to update their response based on the responses of others – repeated until consensus is reached
- SWOT Analysis (BUSINESS)
 - Strengths, Weaknesses, Opportunities and Threats
 - This technique allows finding strengths and weaknesses



SWOT analysis of Google Checkout.

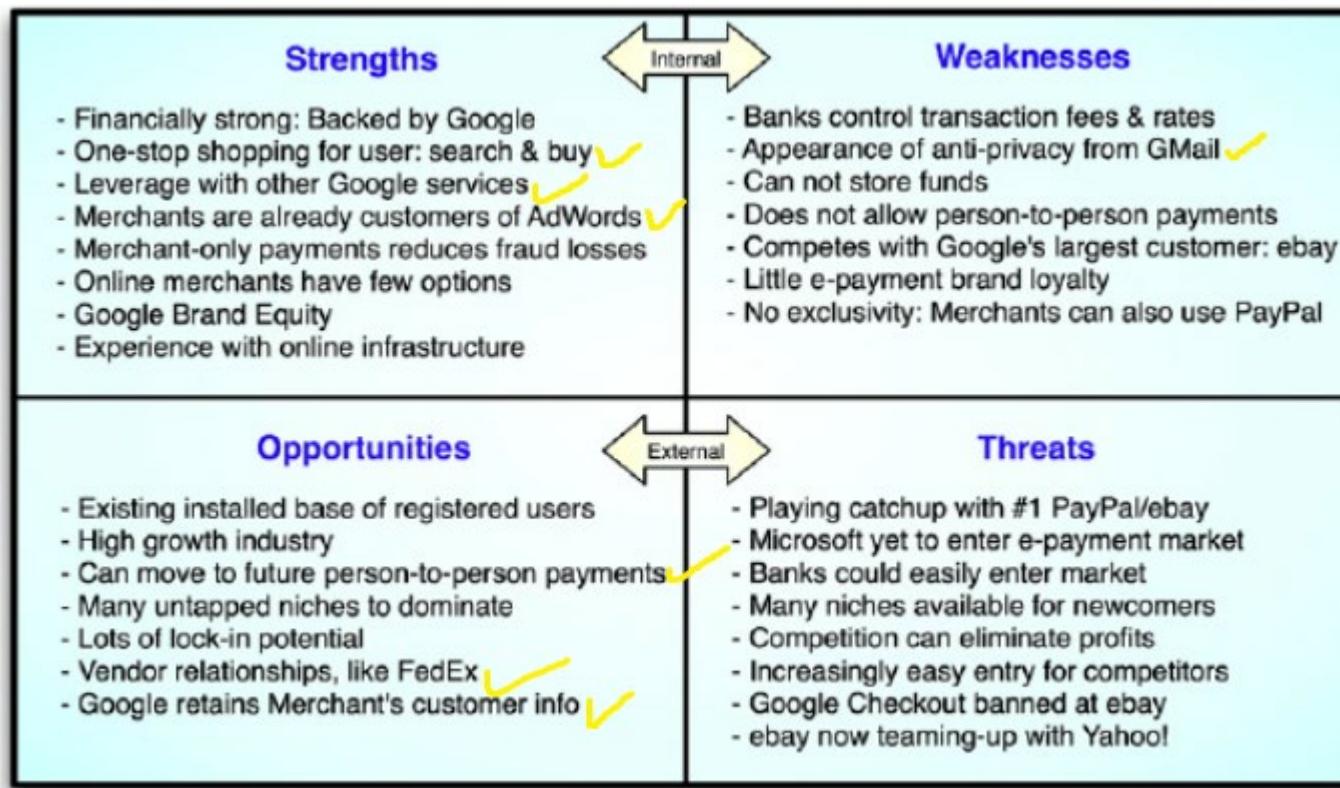


Fenwick, David & Daim, Tugrul & Gerdsri, Nathasit. (2009). Value Driven Technology Road Mapping (VTRM) process integrating decision making and marketing tools: Case of Internet security technologies. Technological Forecasting and Social Change - TECHNOL FORECAST SOC CHANGE. 76. 1055-1077.
10.1016/j.techfore.2009.04.005.



Risk strategy and responses

SWOT analysis of Google Checkout.



Fenwick, David & Daim, Tugrul & Gerdsri, Nathasit. (2009). Value Driven Technology Road Mapping (VTRM) process integrating decision making and marketing tools: Case of Internet security technologies. Technological Forecasting and Social Change - TECHNOL FORECAST SOC CHANGE. 76. 1055-1077.
10.1016/j.techfore.2009.04.005.



MELBOURNE

Four common strategies to handle *threats*:

1. Accept or Ignore
2. Avoid
3. Mitigate
4. Transfer



- Four common strategies to handle *opportunities*:

1. Exploit:

Add work or change the project to make sure the opportunity occurs

2. Enhance:

Increase the probability and positive impact of risk events

3. Share:

Allocate ownership of opportunity to a third-party

4. Accept

This means that we believe that the cost to exploit or enhance is not justifiable so do nothing about it.



We have looked at PayPal and Google Checkout

Another would be -
Alibaba



Two practice timed quiz

Set up a practice timed quiz within a time limit

Any Qs



SWEN90016
Software Processes & Project Management

Risk Management

*Marion Zalk
Department of Computing and Information Systems
The University of Melbourne
mzalk@unimelb.edu.au*



Learning Outcomes

MELBOURNE

1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



What is a risk?

MELBOURNE

- A **risk** is a:

Possible future event that has negative results

Hazard; peril; or exposure to loss or injury

- Webster's dictionary

An uncertain event or condition that, if it occurs, has a positive or negative effect on the project objectives

- PMBOK

- The first two definitions above treat risk and always negative, whereas the third definition considers positive as well as negative impacts - **opportunities** (we will stick with the third definition)



- Risk is different to uncertainty although they are related.
- ***Uncertainty:***
 - Lack of complete certainty about an event/outcome
 - The event/outcome has a probability of less than 1
 - E.g. outcome of a sporting event
- ***Risk:***
 - Uncertainty that has an impact
 - E.g. If you have placed a bet on the sporting event, or have some other personal stake in it, then there is risk associated with the outcome of the sporting event

Risk is a result of uncertainty but not every uncertainty is a risk.



MELBOURNE

- We deal with risks in our lives every day
 - e.g. Planning to get to the lecture
- Projects have many possible risks, that could have significant impacts on the outcomes:
 - Business risks
 - Project risks
 - Product risks
- A planned Risk Management process is essential
- The goal of project risk management is to:
minimising the impact of potential negative risks while maximising the impact of potential positive risks



1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



Risk Management Process

MELBOURNE





- **Plan**
 - How to approach and plan risk management activities?
- **Identify**
 - Identify the possible risks
- **Analyse and Assess (Qualitative and Quantitative):**
 - Identify the relative priorities of the identified risks
- **Respond (Action):**
 - How can we reduce the likelihood or impact of risks?
- **Monitor and Control:**
 - How can we detect the ongoing status of our risks? How can we control them effectively and efficiently?



Learning Outcomes

MELBOURNE

1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



- The output of risk management planning is a ***Risk Management Plan (RMP)*** that documents the procedures for managing risks throughout a project
- The project team should review the RMP and understand and implement the organisation's and the sponsor's approaches to risk management
- The level of detail will vary with the needs of the project



- The Risk Management Plan
 - Methodology
 - Roles and Responsibilities
 - Budget and Schedule
 - Risk Categories
 - Risk Probability and Impact
 - Tracking
 - Risk Documentation
 - Contingency Plans
 - Fall-back Plans



1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



- Determine which events should be considered as risks by analysing the following:
 - Is the *probability* of the event occurring greater than *zero*?
 - What is the *impact* of the event on the project?
 - Do we have some *degree of control* over the event or its outcome?
- Generic Risks:
 - Threats or opportunities common to every software project (e.g. staff turnover, budget and schedule pressures)
- Product-specific Risks:
 - Threats or opportunities specific to the product, and can only be identified by people who have a clear understanding of the product and technology



- **Project risks**
 - Affect the planning of the project
 - e.g. Budget, Schedule, Scope, Personnel, etc.
- **Product risks**
 - Affect the quality or performance of the outcome being developed
 - e.g. Design problems, implementation problems, interface problems, maintenance problems, verification problems
- **Business risks**
 - Affect the economic success of the project
 - e.g. No demand for product, loss of management support, loss of external funding for the project etc.



- **Risk identification**

- Deals with using a systematic approach for identifying and creating a list of threats and opportunities that may impact the project's goals

- **Risk identification techniques**

- Pondering
- Interviewing
- Brainstorming
- Checklists
- Delphi Technique
- SWOT Analysis



- Pondering
 - This simply involves an individual taking the “pencil and paper” approach of risk identification, which involves sitting and thinking about the possible risks that could occur in the project
 - This is one of the initial risk assessment tasks used in many projects
- Interviews/questionnaires
 - Interviewing project stakeholders, or asking them to fill out questionnaires, to harness their knowledge of a domain
 - It is unlikely that a risk manager in a software project will have sufficient knowledge of the methods and tools to be employed to provide a comprehensive view of the risks, so input from stakeholder and domain experts is essential



- Brainstorming
 - The team can use a *risk framework* or the *Work Breakdown Structure (WBS)* to identify threats and opportunities
 - The key is to encourage contributions from everybody
 - The group then discuss and evaluate
- Checklists
 - This involves the use of standard checklists of possible risk drivers that are collated from experience
 - These checklists are used as triggers for experts to think about the possible types of risks in that area



- Delphi Technique
 - A group of experts are asked to identify risks and their impact
 - The responses are then made available to each other anonymously
 - The experts are then asked to update their response based on the responses of others – repeated until consensus is reached
- SWOT Analysis
 - Strengths, Weaknesses, Opportunities and Threats
 - This technique allows finding strengths and weaknesses as well



MELBOURNE

- Example: Risk of a third-party software application

Consider the example of using a third-party software application to provide some functionality of a system that is being developed. The third-party application is developed in parallel with the system:
- Risks:
 1. The application could be delivered later than planned, thereby delaying the delivery of the entire system.
 2. Once complete, the third-party application may not be reliable enough to be used, meaning that a new third-party application providing the functionality will need to be sourced or developed.
 3. The third-party application may deliver behaviour that is inconsistent with the expectations of the system developers, meaning that a new third-party application providing the functionality will need to be sourced or developed.



Identified Risks - example

MELBOURNE

Risk Source Category	Possible Risk Examples /Risk Factors
Project Size and Complexity	<ul style="list-style-type: none">• Effort Hours• Calendar Time• Estimated Budget• Team and Size (Number of Resources)• Number of Sites• Number of Business Units• Number of Dependencies on other Projects• Degree of Business Change
Requirements	<ul style="list-style-type: none">• Volatile Requirements• Unrealistic Quality Requirements• Complex Requirements
Change Impact	<ul style="list-style-type: none">• Replacement of New System• Impact on Business Policies• Impact on Organisational Structure• Impact on Systems Operations



Identified Risks - example

MELBOURNE

Risk Source Category	Possible Risk Examples /Risk Factors
Stakeholders	<ul style="list-style-type: none">• All key stakeholders have not been identified• Missing "Buy-In" from a key stakeholder• Stakeholder needs not completely identified• Key stakeholders not fully engaged
Organization	<ul style="list-style-type: none">• Changes to Project Objectives• Lack of Priorities• Lack of Project Management "Buy-In" and Support• Inadequate Project Funding• Misallocation and Mismanagement of Resources
Scope	<ul style="list-style-type: none">• Grope• Leap• Creep
Schedule	<ul style="list-style-type: none">• Estimated Assumptions are Not Holding True• Scheduled Contingency is Not Adequate• Inadequate or Poor Estimation



Lecture Break

BREAK

Please return promptly as the

Lecture will re-start in *5 mins*



|



MELBOURNE



Risk Planning → **Risk Management Plan**

Risk Identification:

Kinds of Risks:

Project
Product
Business

Identification Techniques:

Pondering	Interviewing
Brainstorming	Checklists
Delphi	SWOT Analysis



Learning Outcomes

MELBOURNE

1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



- Risk analysis and assessment provide a systematic approach for evaluating the risks
- **Risk analysis**
 - Identify each identified risk's *probability* and *impact*
- **Risk assessment**
 - *Prioritize* risks so that an effective *risk strategy* can be formulated
- Two approaches for analysis and assessment:
 - Qualitative: subjective assessment based on experience/intuition
 - Quantitative: mathematical and statistical techniques



- The important steps of risk analysis are:
 1. Estimating the *risk probability (P)*
 - this is an estimation of the probability that the risk will occur
 - usually done based on expert judgement
 2. Estimating the *risk impact (I)*
 - the impact that the risk will have on the project
 - Usually measured in a scale of 1 – 5 (or 10):
 - (1)no impact; (2) minimal impact; (3) moderate impact; (4) severe impact; and (5) catastrophic impact
 - Impact can be expressed as a monitory value



- The important steps of risk analysis cont..

3. Compute *risk exposure (or P *I Score)*

$$\text{Riskexposure} = P * I$$

4. Identifying the root cause

- It is important that one identifies the root causes of all risks
- If this root cause can be identified, then all of these risks can be controlled by addressing the root cause



- Example: Risk of a third-party software application

Consider the example of using a third-party software application to provide some functionality of a system that is being developed. The third-party application is developed in parallel with the system:
- Risks:
 1. The application could be delivered later than planned, thereby delaying the delivery of the entire system.
 2. Once complete, the third-party application may not be reliable enough to be used, meaning that a new third-party application providing the functionality will need to be sourced or developed.
 3. The third-party application may deliver behaviour that is inconsistent with the expectations of the system developers, meaning that a new third-party application providing the functionality will need to be sourced or developed.



MELBOURNE

Risk ID	Risk	Probability (0-1)	Impact	Exposure
1	The application could be delivered later than planned, thereby delaying the delivery of the entire system.	0.15	\$10,000	\$1500
2	Once complete, the third-party application may not be reliable enough to be used, meaning that a new third-party application providing the functionality will need to be sourced or developed	0.05	\$20,000	\$1000
3	The third-party application may deliver behaviour that is inconsistent with the expectations of the system developers, meaning that a new third-party application providing the functionality will need to be sourced or developed	0.2	\$20,000	\$4000

Risk Impact Analysis Table



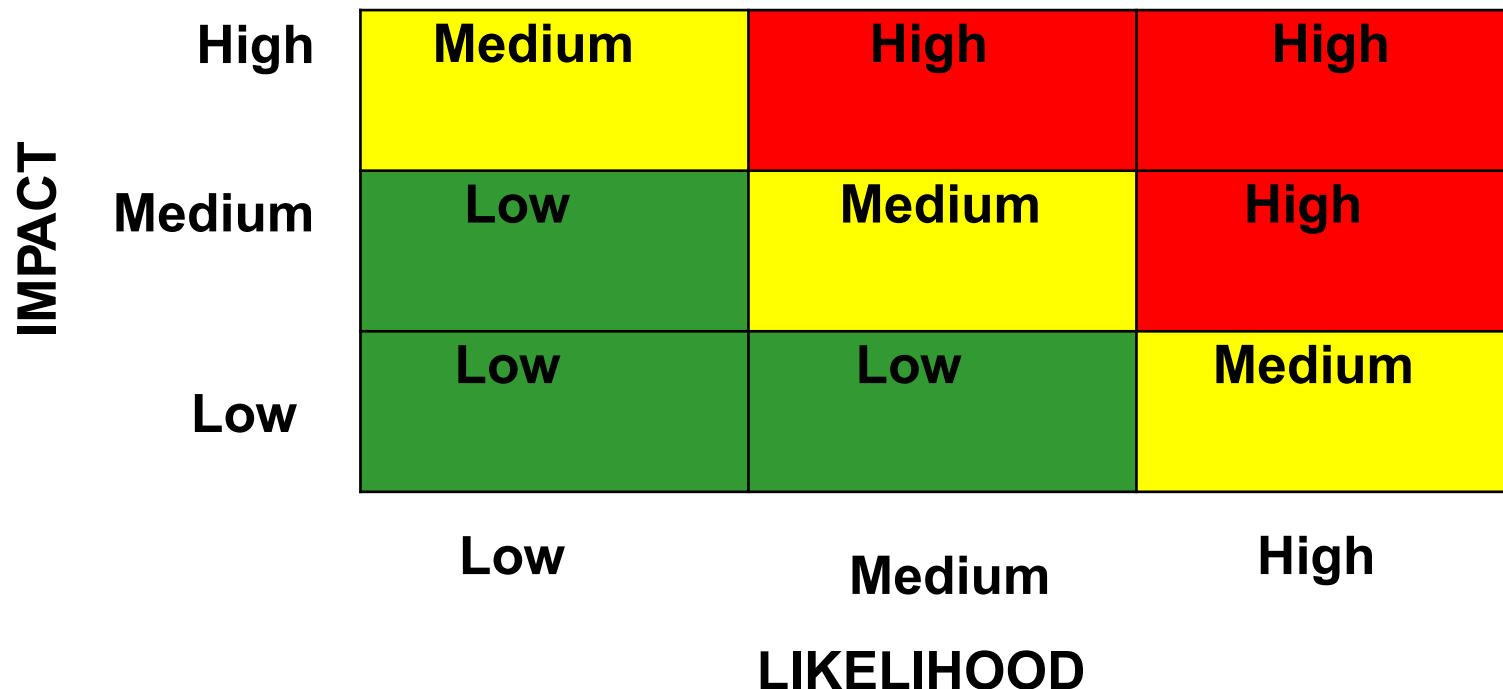
MELBOURNE

Risk ID	Risk	Probability (0 – 100%)	Impact (0 – 10)	Exposure	Rank
1	A key member leaving the project	40%	4	1.6	4
2	Client unable to define scope and requirements	50%	6	3.0	3
3	Client experiences financial problems	10%	9	0.9	5
4	Response time not acceptable to the user/client	80%	6	4.8	1
5	Technology does not integrate with existing application	60%	7	4.2	2
6	Financial manager deflects resources away from the project	20%	3	0.6	6
7	Client unable to obtain license agreement	5%	7	0.4	7

Risk Impact Analysis Table

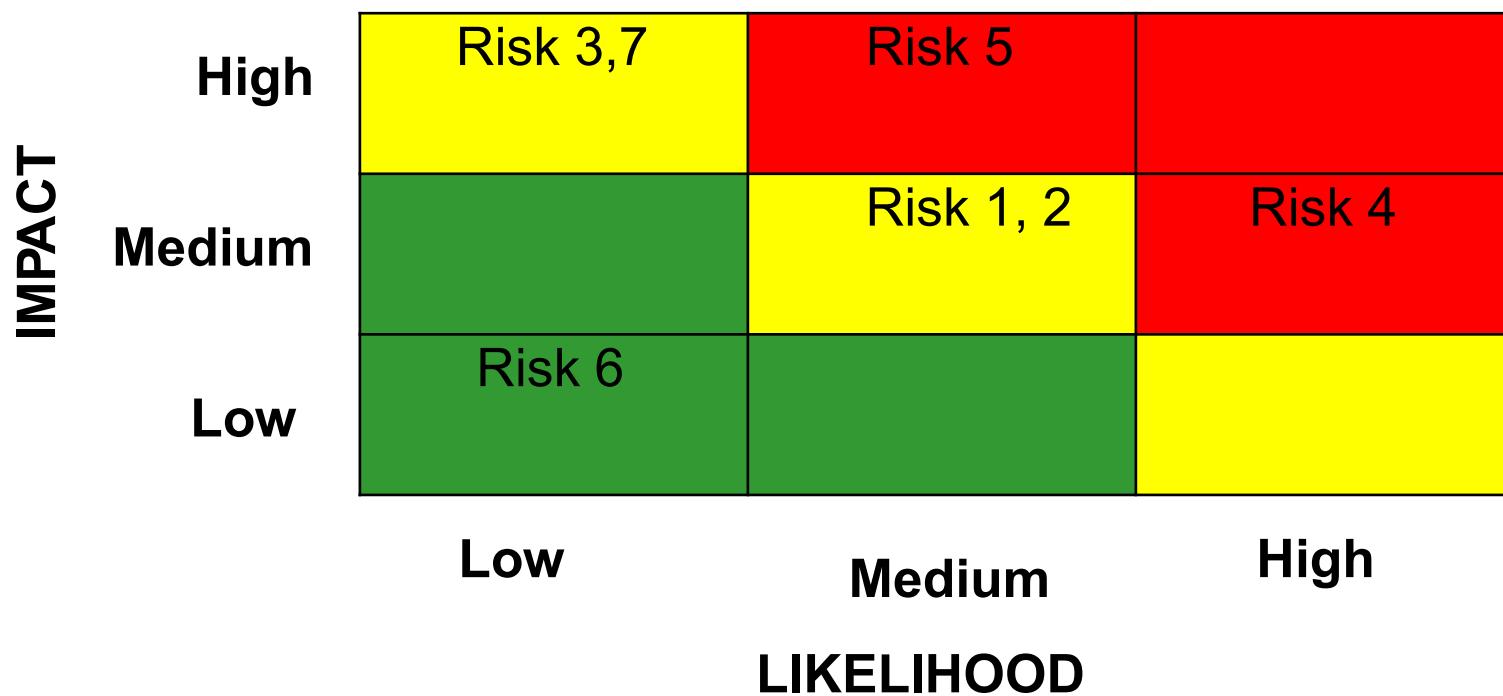


- **Risk matrix** - define the level of risk by considering the probability or likelihood consequence severity.
- A mechanism to increase visibility of risks and assist management decision making.





MELBOURNE





MELBOURNE

- Quantitative approaches include mathematical and statistical techniques
- They are based on modelling a particular risk situation - probability distributions of risks are the main consideration
- Common Techniques:
 - Decision Tree Analysis
 - Simulation
 - Sensitivity Analysis
- Quantitative approaches are beyond the scope of this subject



Learning Outcomes

MELBOURNE

1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (**risk strategies**)
 - monitor and control risks



- Purpose of risk analysis and assessment is to identify what opportunities and threats *should be addressed*
- It is not feasible (or advisable) to respond to every threat or opportunity because this requires *resources*, which are usually diverted from the project, which could have more negative impacts on the project
- Therefore, it is important to select appropriate response strategies



- Four common strategies to handle *threats*:

1. Accept or Ignore

This means that we believe that the risk is of an acceptable exposure, that we hope that the event does not occur, or that the risk exposure is less than the cost of any techniques to avoid, mitigate, or transfer it.

2. Avoid

This means that we completely prevent the risky event from occurring, by either ensuring its probability is 0, or ensuring its impact 0.



- Four common strategies cont..

3. Mitigate

This involves employing techniques to reduce the probability of the risk, or reduce the impact of the risk. This results in a residual risk — that is, a risk consisting of the same event, but with a lower probability/impact, and therefore low exposure. We then must analyse the residual risk as we would our primary risk.

4. Transfer

This involves transferring the burden of the risk to another party. Insurance is one example of risk transfer, in which the impact of the risk is offset by payments from the insurer. Another example is outsourcing a portion of the work to somebody with more knowledge and expertise, which comes at a cost.



MELBOURNE

- Example: Risk of a third-party software application

Consider the example of using a third-party software application to provide some functionality of a system that is being developed.

Strategy	Response
Ignore	Do nothing because the vendor is reliable and have delivered quality software in the past.
Avoid	Developing the required functionality in house, rather than buying it or change the requirements so that the functionality is not required at all.
Mitigate	Make the request date well before the required date. We can also reduce the impact of the risk by designing the system such that the third-party application is accessed via a standard interface, and by producing a dummy implementation of that interface that allows development to continue if the third-party application is delivered late.
Transfer	Specifying in the contract that any costs resulting from late delivery of the system will be paid for by the vendor of the third-party application.



- Four common strategies to handle *opportunities*:

1. Exploit:

Add work or change the project to make sure the opportunity occurs

2. Enhance:

Increase the probability and positive impact of risk events

3. Share:

Allocate ownership of opportunity to a third-party

4. Accept

This means that we believe that the cost to exploit or enhance is not justifiable so do nothing about it.



- Once risks and strategies are identified, they can be documented as a part of a risk response plan, also called a Risk Register.
- Template of a simple risk register
 - Risk ID: a unique identification for the risk
 - Trigger: the trigger that flags that the risk has occurred
 - Owner: the person or group responsible for monitoring and responding
 - Response: the strategy for responding
 - Resources: required resources

Risk ID	Trigger	Owner	Response	Resources Required

Risk Register



1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



- Once the risk response plan has been created, triggers must be monitored to keep track of various project risks
- New threats and opportunities may arise in the course of the project – they must be identified, analysed and responded to
- Risk monitoring must be part of the overall monitoring and control of the project



- Tools for monitoring and controlling:
 - Risk Audits:
 - external team looks at comprehensiveness of the identification process and ensuring other procedures and processes are in place
 - Risk Reviews:
 - internal reviews of risks periodically that result in status reports generated for PM and those who need-to-know
 - Risk status meetings:
 - risks must be reviewed and discussed in project status meetings, which are periodically held in projects (e.g. weekly meetings)



Risk Management Process

MELBOURNE





1. Understand the fundamentals of risk management
2. Understand the Risk Management Process
3. Understand how to:
 - plan risk management activities
 - identify risks
 - analyze and assess risks
 - respond to risks (risk strategies)
 - monitor and control risks



- Shari L. Pfleeger and Joanne M. Atlee. Software Engineering: Theory and Practice. Prentice–Hall International, 3rd edition, 2006.
- R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
- J.T. Marchewka. Information Technology Project Management. John Wiley & Sons, fourth edition, 2012.



SWEN90016

Software Processes & Project Management

*Department of Computing and Information Systems
The University of Melbourne*

Copyright University of Melbourne 2017

2021 – Semester 1
Lecture 9

Intended Learning Objectives

Ethics and Australian Computer Society Code Of Ethics.

Outsourcing.

Procurement.

Contracts.



L9.1 - Ethics





Why should organisations be ethical?

L9.1 - Ethics



Ethics – What is it?

- *Organisational ethics express the values of an organization to its employees and/or other entities irrespective of governmental and/or regulatory laws.*
- *Ethics are the principles and values used by an individual to govern his or her actions and decisions*

Ethics

Ethics in organisations are important because:



- **Satisfies Basic Human Needs:** Being fair, honest and ethical is one the basic human needs. Every employee desires to be such himself and to work for an organization that is fair and ethical in its practices.
- **Creates Credibility:** An organisation that is believed to be driven by moral values is respected in the society.
- **Unites People and Leadership:** An organisation driven by values is revered by its employees also. They are the common thread that link all employees regardless of position.
- **Set the basis for Decision Making:**
- **Long Term Gains:** Organisations guided by ethics and values last and are profitable in the long run.

WILLING TO LEARN

Ethics

Ethics are not only for the 'big issues'

- Should we execute criminals?
- Can we destroy embryos for medical research?
- Lie under oath?



They inform our day-to-day interactions:

- How we treat our fellow colleagues.
- What information / resources can we use or take from work?
- Should we tell a work colleague a truth even though we know it will upset them?

WILLING TO LEARN

Ethics

Questions to ask & consider before making a decision:

1. Would I be happy for this decision to be headlining the news tomorrow or be confronted with this in my work / friendship group?
2. Is there a universal rule that applies here?
3. Will the proposed course of action bring about a good result?
4. What would happen if everybody did this?
5. What will this proposed action do to my character or the character of my organisation?
6. Is the proposed course of action consistent with my values and principles?



Reference <http://www.ethics.org.au/about/what-is-ethics>

Ethics – Your personal beliefs

An example:

Modern slavery encompasses slavery, servitude, the worst forms of child labour, forced labour, human trafficking, debt bondage, slavery like practices, forced marriage and deceptive recruiting for labour or services.

Many Australian businesses may be unaware of the risk that they have slavery in their business or supply chains. Statistically, the incidence of modern slavery within Australia appears to be relatively low, but it is likely that the statistics reflect a low level of awareness of the issues, and the actual incidence may be much higher, both domestically and overseas.

As at 2018, the Walk Free Foundation's Global Slavery Index estimated:

- In excess of 40 million people globally are subject to some form of modern slavery and collectively approximately US\$150 billion per year is generated in the global private economy from forced labour alone;
- ~25 million people (79% females) in Asia-Pacific Region are 'enslaved' (62 per cent of all people enslaved); and
- 15,000 people in Australia are enslaved.



WILLING TO LEARN

Ethics

THERE IS
NO RIGHT WAY
TO DO
A WRONG THING.





WILLING TO LEARN

Australian Computer Society Code Of Ethics

1. The Primacy of Public Interest.

- *You will place the interests of the public above those of personal, business or sectional interests*

2. The Enhancement of Quality of Life.

- *You will strive to enhance the quality of life of those affected by your work*

3. Honest.

- *You will be honest in your representation of skills, knowledge, services & products.*

4. Competence.

- *You will work competently and diligently for your stakeholders*

5. Professional Development.

- *You will enhance your own professional development, your colleagues & staff.*

6. Professionalism.

- *You will enhance the integrity of the ACS & the respect of its members for each other.*

References: www.acs.org.au/content/dam/acs/rules-and-regulations/Code-of-Ethics.pdf
www.acs.org.au/content/dam/acs/rules-and-regulations/Code-of-Professional-Conduct_v2.1.pdf



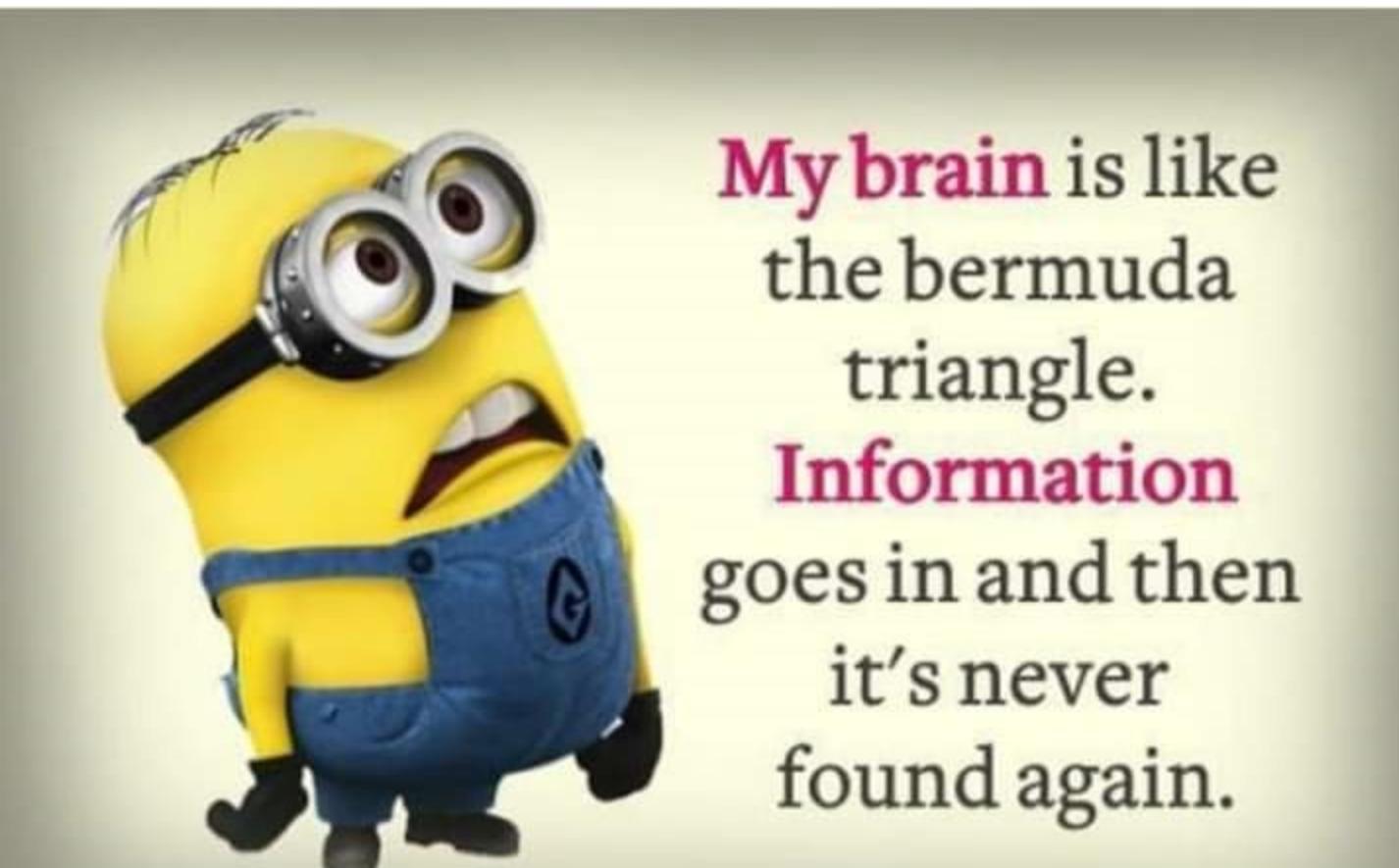
Question added for Assignment 1

THERE IS
NO RIGHT WAY
TO DO
A WRONG THING.





Revision 7:45pm tonight



Minion Quotes (2021, 01 May) Minion Quotes <http://www.facebook.com/Minion Quotes>



SWEN90016
Software Processes & Project Management

Project Scheduling

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

Copyright University of Melbourne 2018

2021 – Semester 1
Lecture 5



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



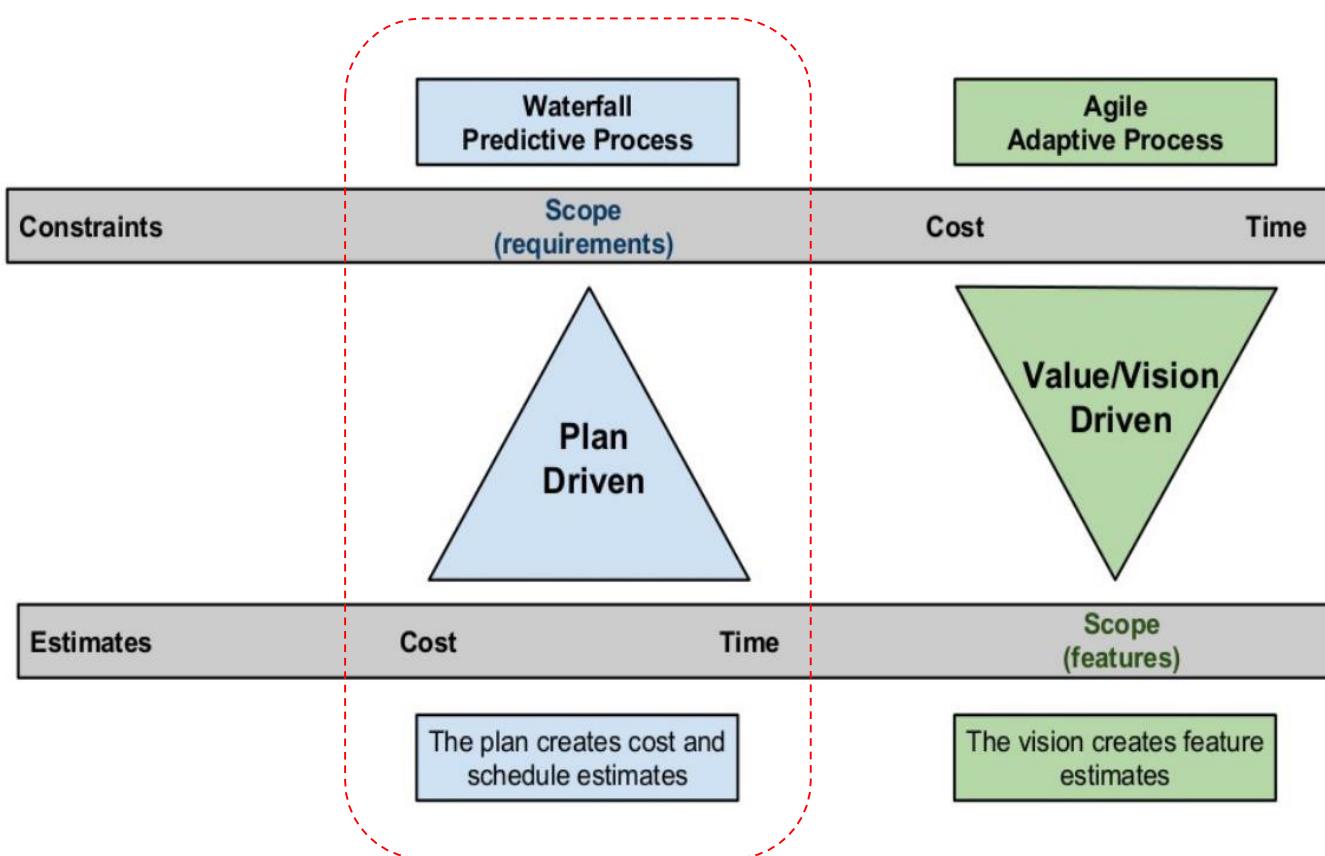
Project Schedule

MELBOURNE

- **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

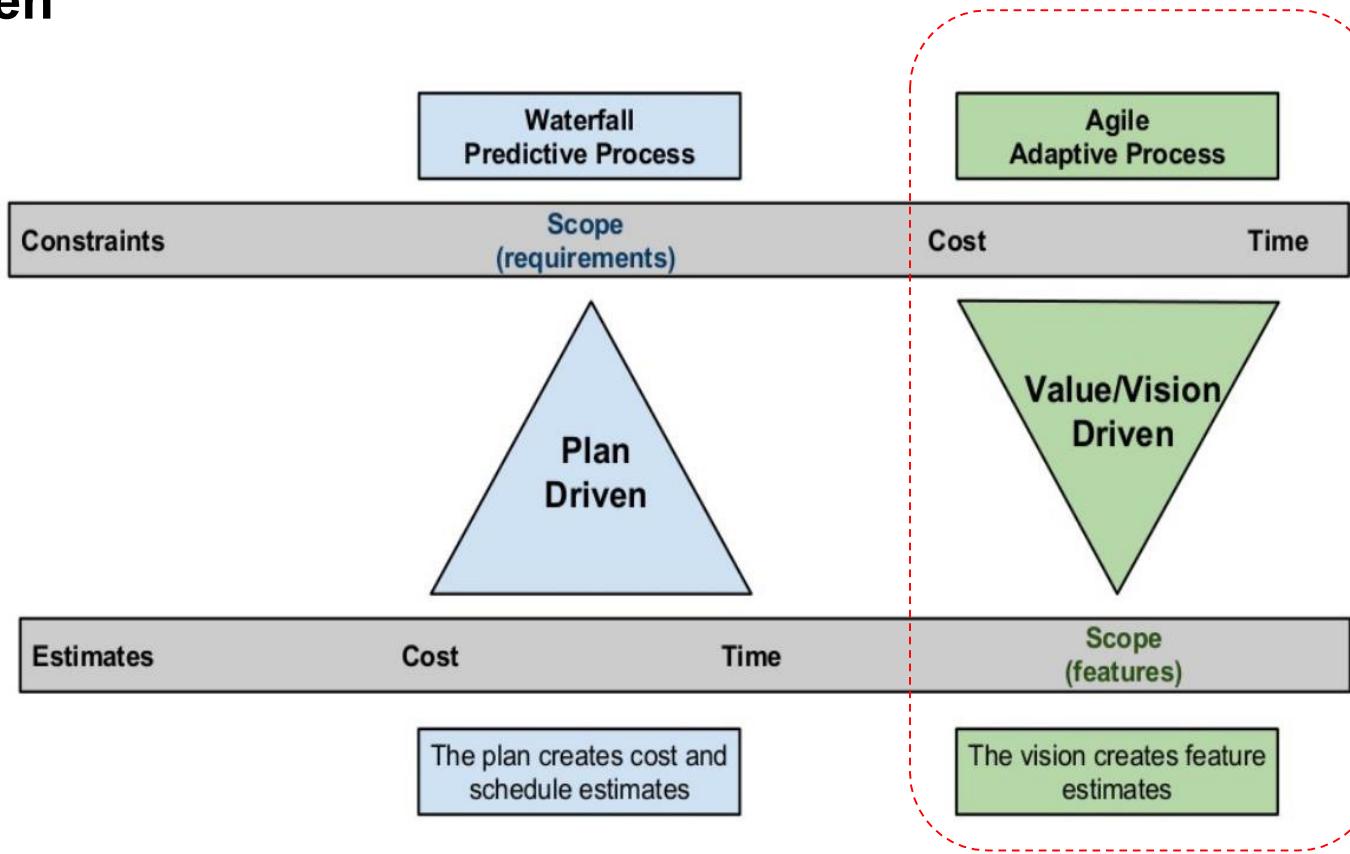


Project planning and scheduling introduced in this topic apply to formal SDLC processes – Plan Driven





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)



MELBOURNE

Redecorate Room

Prepare materials

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

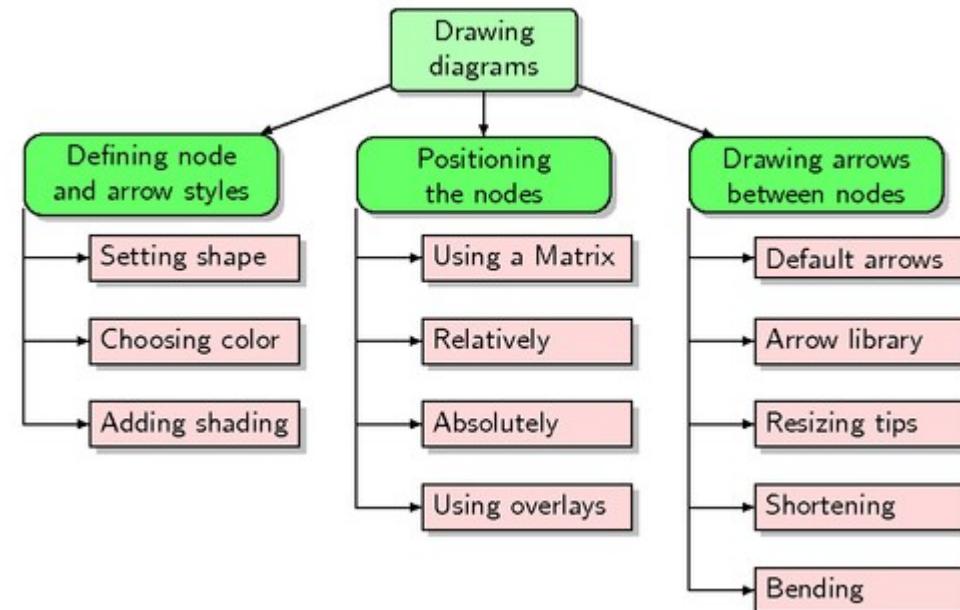
Prepare room

- 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



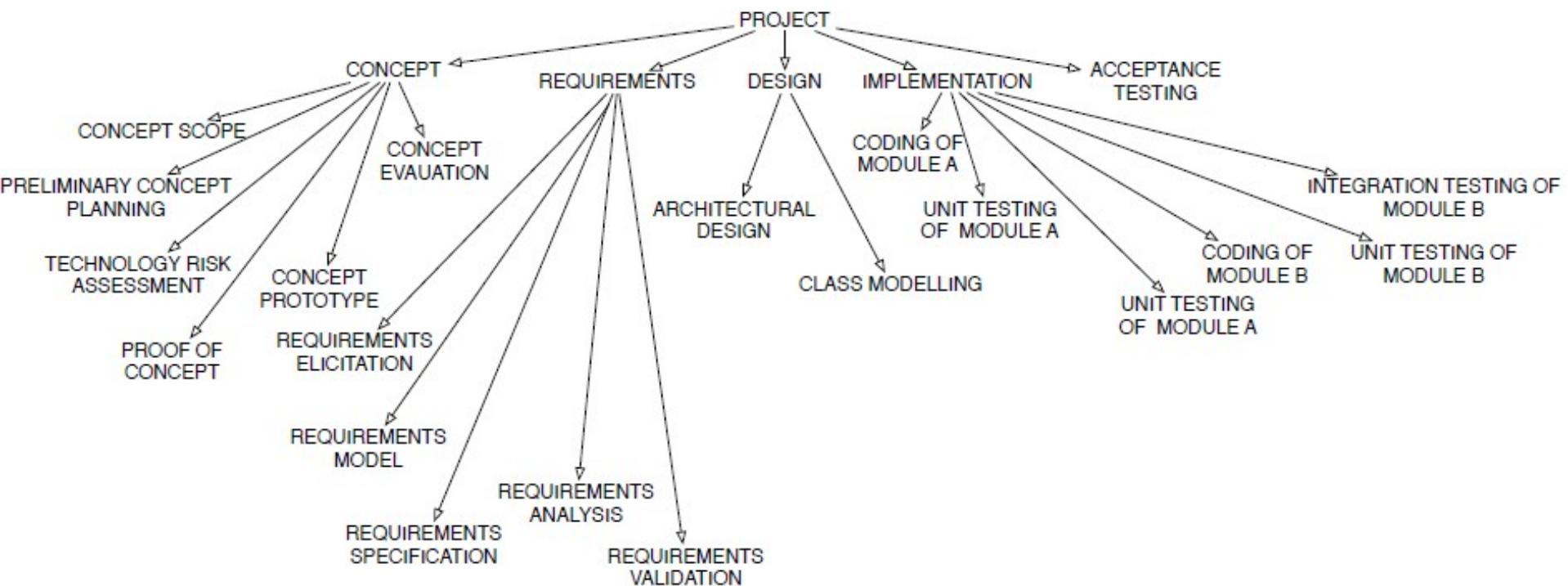
<http://texexample.net/tikz/examples/work-breakdown-structure/>

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



Example – WBS (Software Project)

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

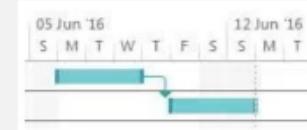
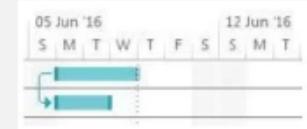
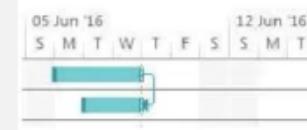
4. Allocate resources for tasks and validate effort

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

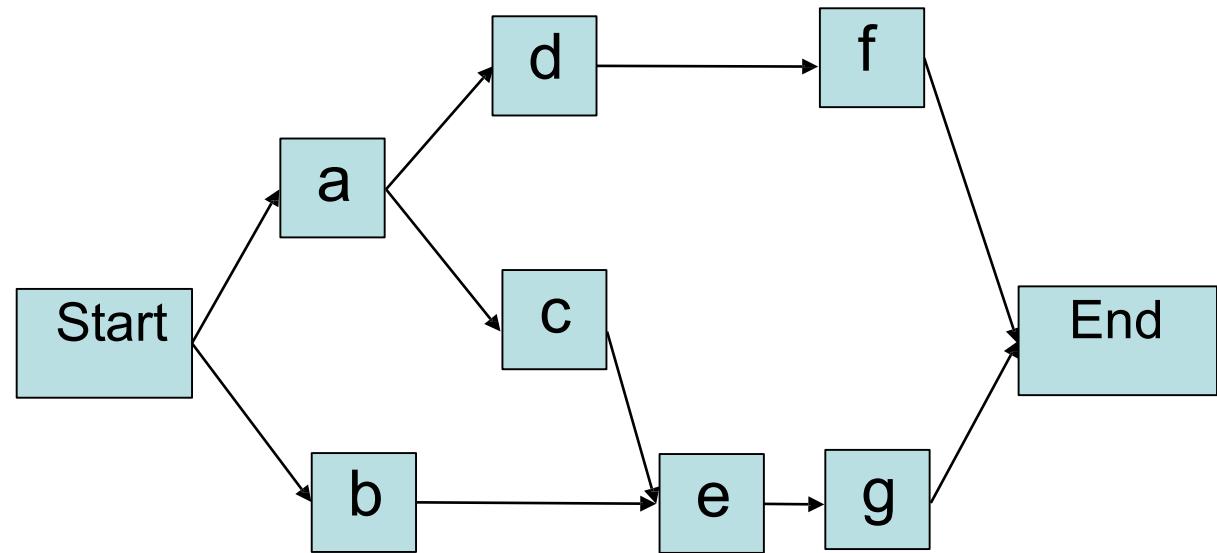


Dependency	Description	Representation
Finish-to-Start	Predecessor must finish before Successor can start	
Start-to-Start	Predecessor must start before Successor can start	
Finish-to-Finish	Predecessor must finish before the Successor can Finish	
Start-to-Finish	Predecessor must start before the Successor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

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



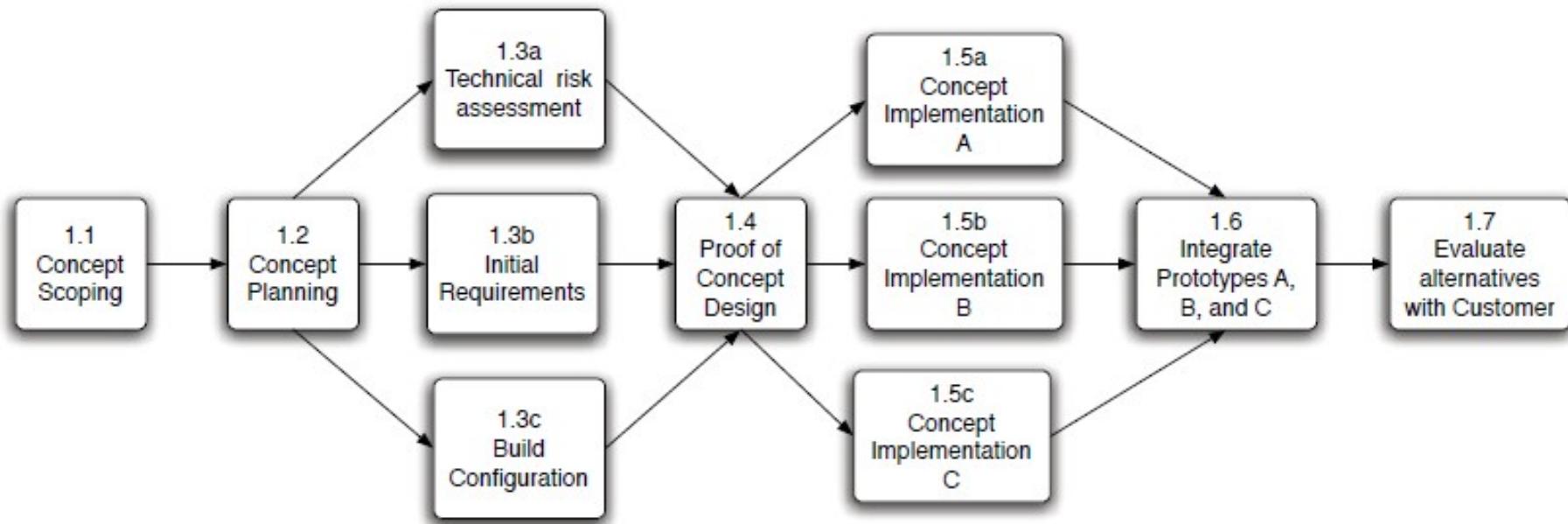
Activity	Predecessor
a	—
b	—
c	a
d	a
e	b, c
f	d
g	e





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

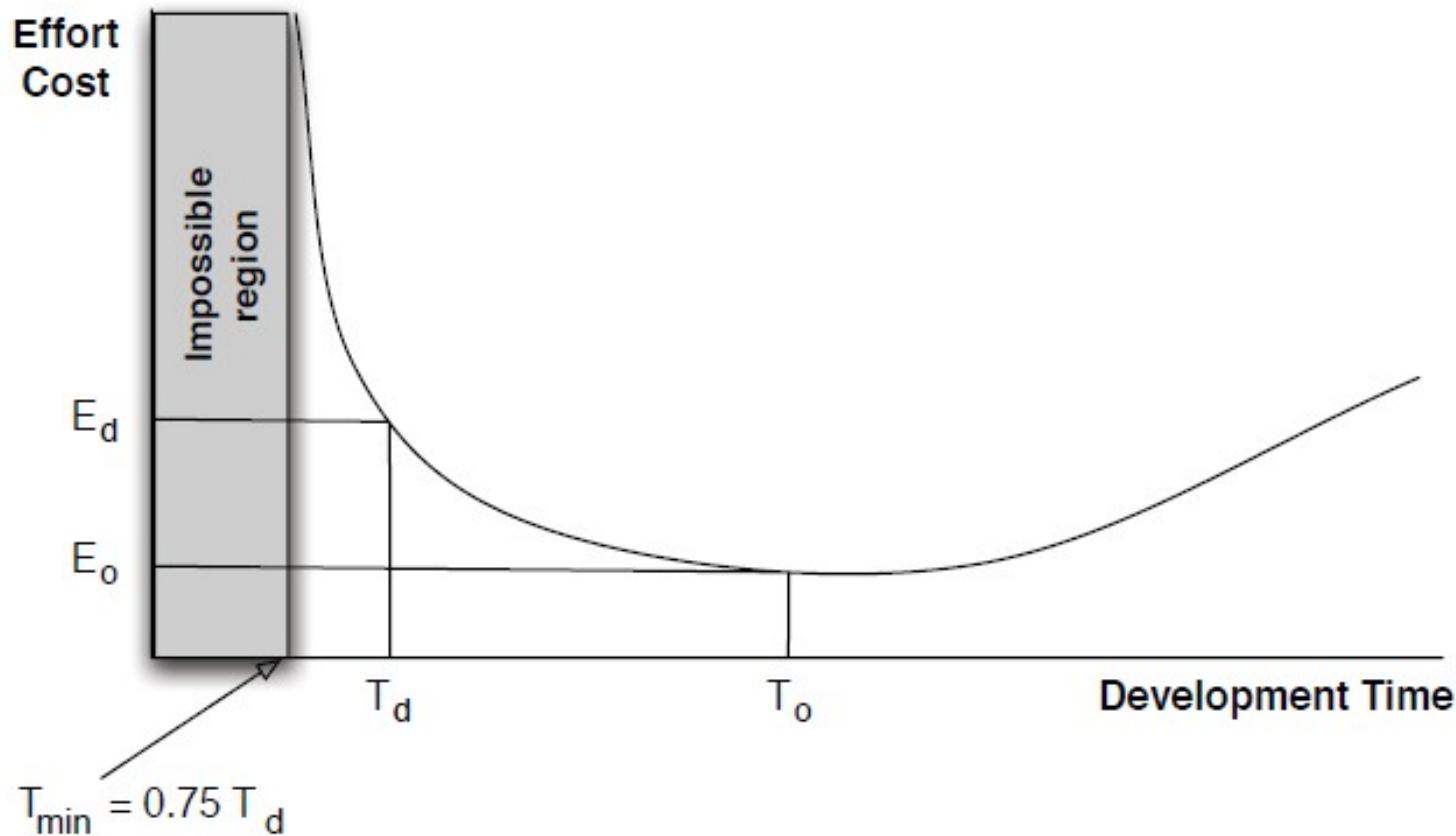




1. Breakdown the task into small chunks you can deal with –
Work Breakdown Structure (WBS)
2. Identify the interdependencies between the broken down tasks and develop a task network
3. Estimate the **effort** and the **time allocation** for each task
4. Allocate resources for tasks and validate effort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



- 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



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	a	4	5	7	5.17
d	a	4	6	10	6.33
e	b, c	4	5	7	5.17
f	d	3	4	8	4.50
g	e	3	5	8	5.17



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



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**



Break- 5 minutes



<https://pmstudycircle.com/2012/04/can-i-take-break-during-my-pmp-certification-exam/>



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**



- **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*



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**

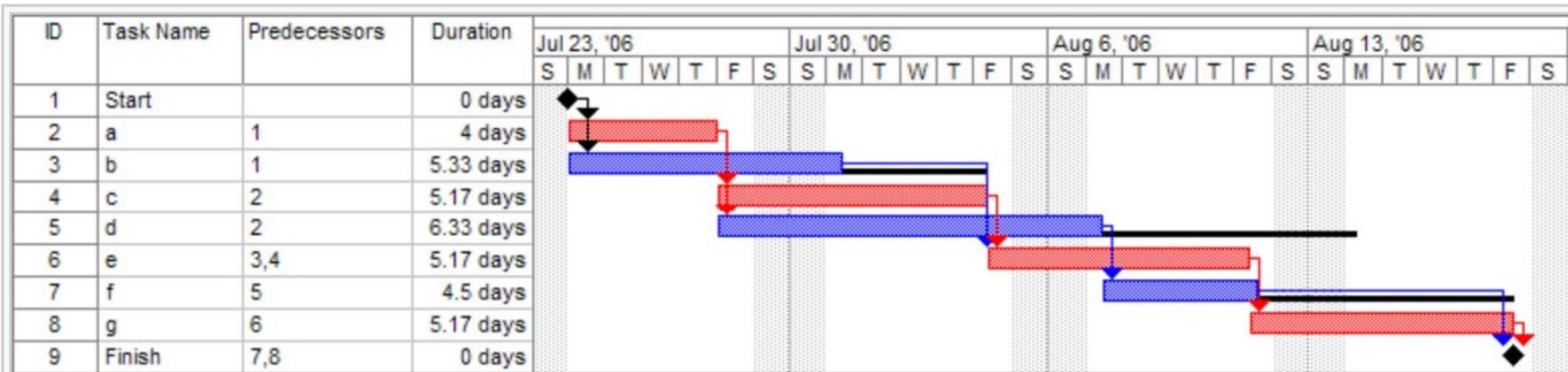
- Mark specific points along a project timeline
- These points may signal anchors such as:
 - a project start and end date
 - a need for external review
 - start and end of a phase
 - a completion of a deliverable

- **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



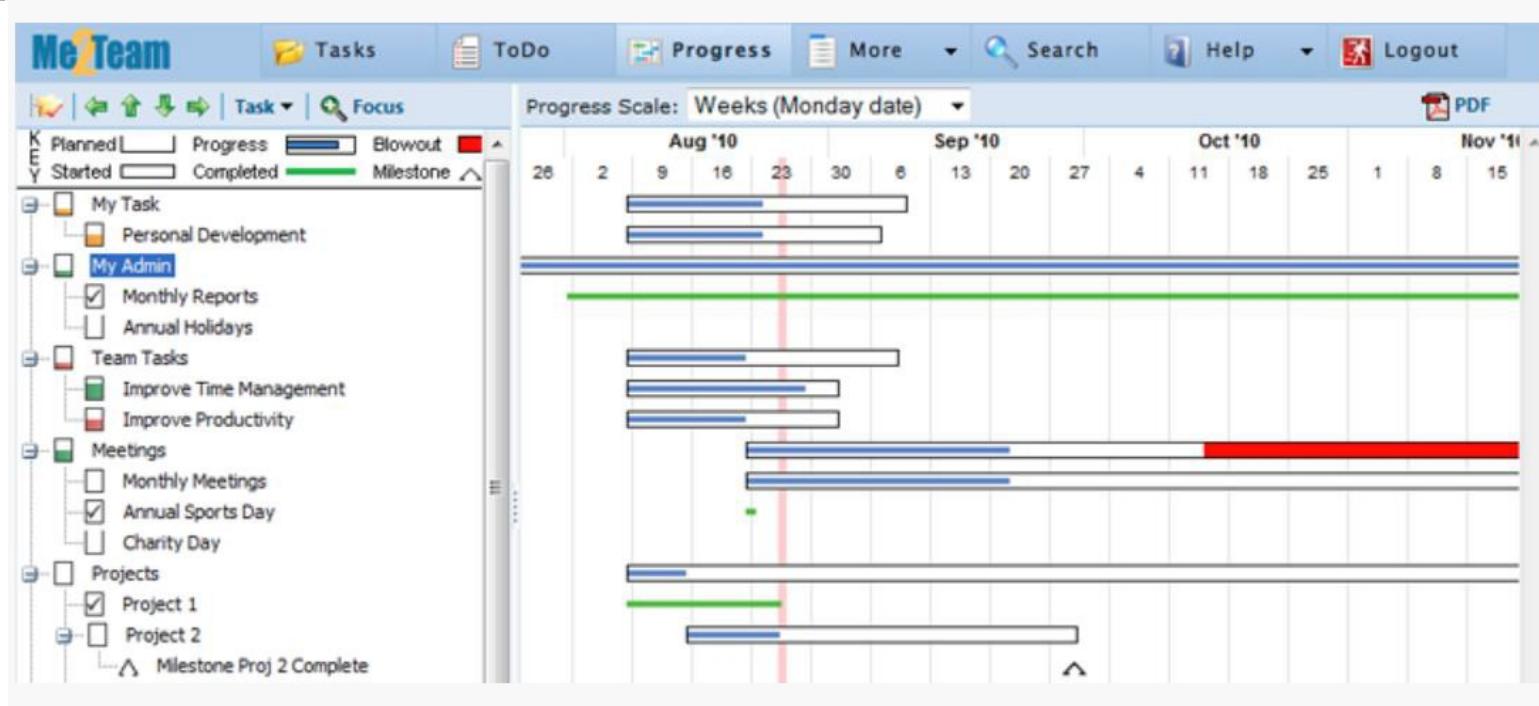
- Was introduced by Henry Gantt in 1910
- Gantt chart is a horizontal bar chart which shows tasks against a timeline – **project schedule**
- Can be used to view planned activities vs progress and therefore is a useful tool for monitoring project progress



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



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



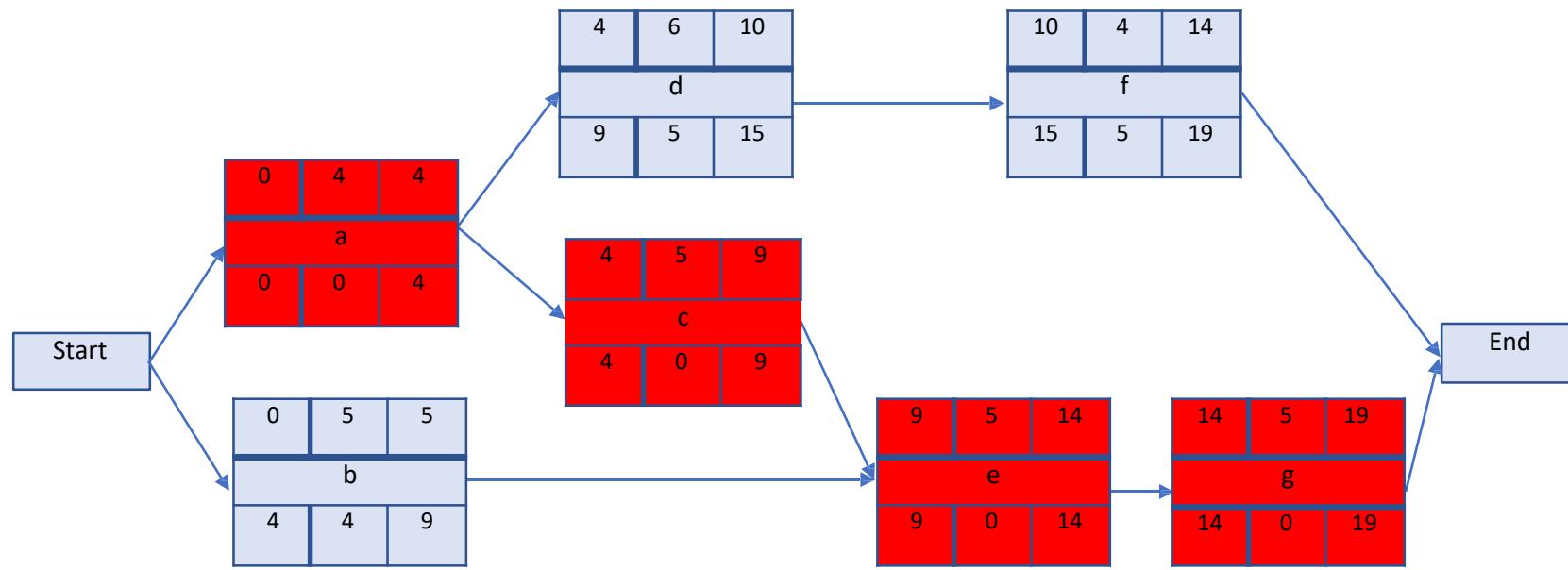
- Involves calculating the following estimates:
 - Earliest start time (ES)
 - Latest start time (LS)
 - Earliest finish time (EF)
 - Latest finish time (LF)
 - Slack time

ES	Duration	EF
Task Name		
LS	Slack	LF



PERT Chart - Example

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

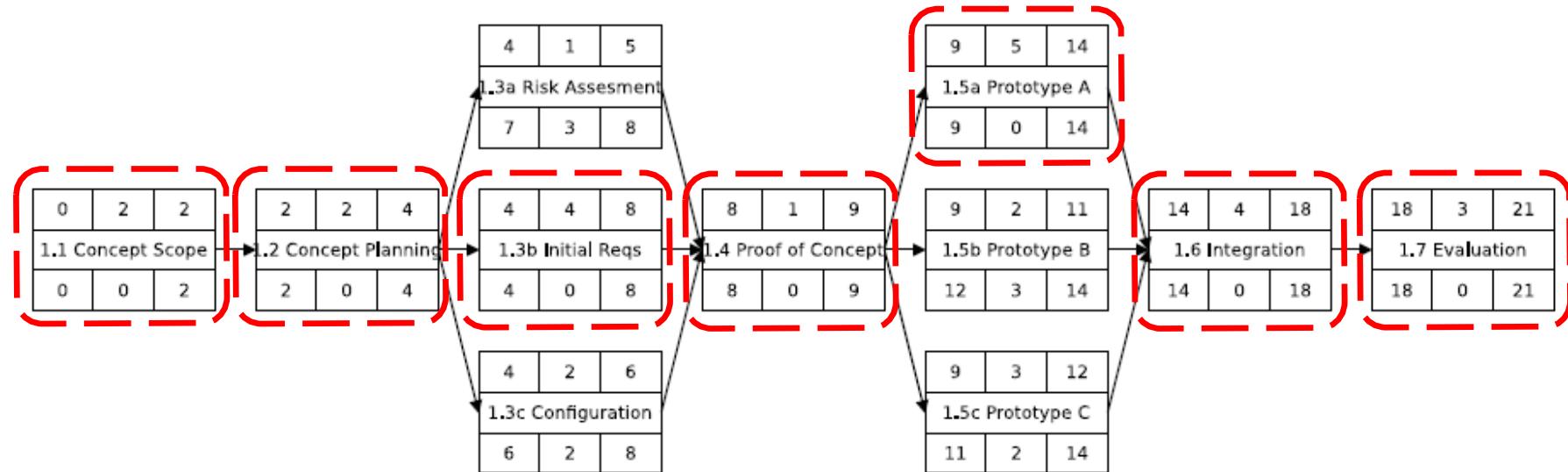


Task	Dependencies	Most Likely Time
1.1 Concept Scoping		2 days
1.2 Concept Planning	1.1	2 days
1.3a Technology Risk Assessment	1.2	1 day
1.3b Initial Requirements	1.2	4 days
1.3c Configuration	1.2	2 days
1.4 Proof of Concept	1.3a, 1.3b, 1.3c	1 day
1.5a Concept Prototype A	1.4	5 days
1.5a Concept Prototype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



- **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



MELBOURNE

Atlassian (Jira), Microsoft project

Product	Rating	Price	Platforms	Deployments	Business Size	
smartsheet	Smartsheet ★★★★★ (395)	\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Mavenlink	Mavenlink ★★★★★ (224)	\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
workzone	Workzone ★★★★★ (38)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
inMotion	inMotion ★★★★★ (32)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Accelo	Accelo ★★★★★ (3)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
monday.com (formerly dapulse)	monday.com ★★★★★ (606)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Workfront	Workfront ★★★★★ (425)	\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Freshservice	Freshservice ★★★★★ (341)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Wrike	Wrike ★★★★★ (745)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	S M L	Visit Website
Airtable	Airtable ★★★★★ (162)	\$\$\$\$\$\$	mac, windows, android	cloud, desktop	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



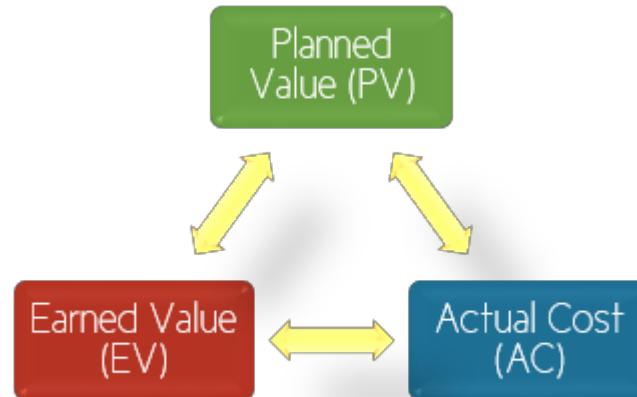
- **How do software projects fall behind schedule?**
One day at a time
 - Fred Brooks, the well-known author of the seminal article *Mythical Man-Months*
- Project scheduling is important, but ***tracking and controlling*** are even more important!



- 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*



- 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



EVA - Example

MELBOURNE

- 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?



EVA - Example

MELBOURNE

- 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



MELBOURNE

- **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**

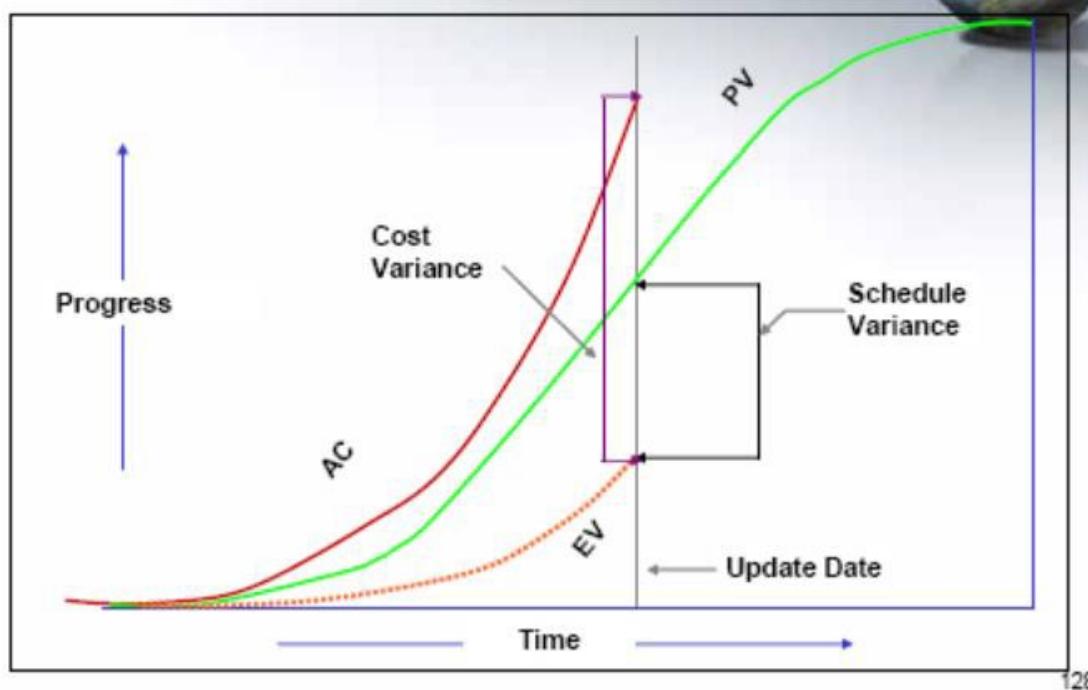
$$\begin{aligned} CV &= EV - AC \\ &= 20,000 - 35,000 \\ &= (15,000) \end{aligned}$$

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

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



Graphic Performance Report



128

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



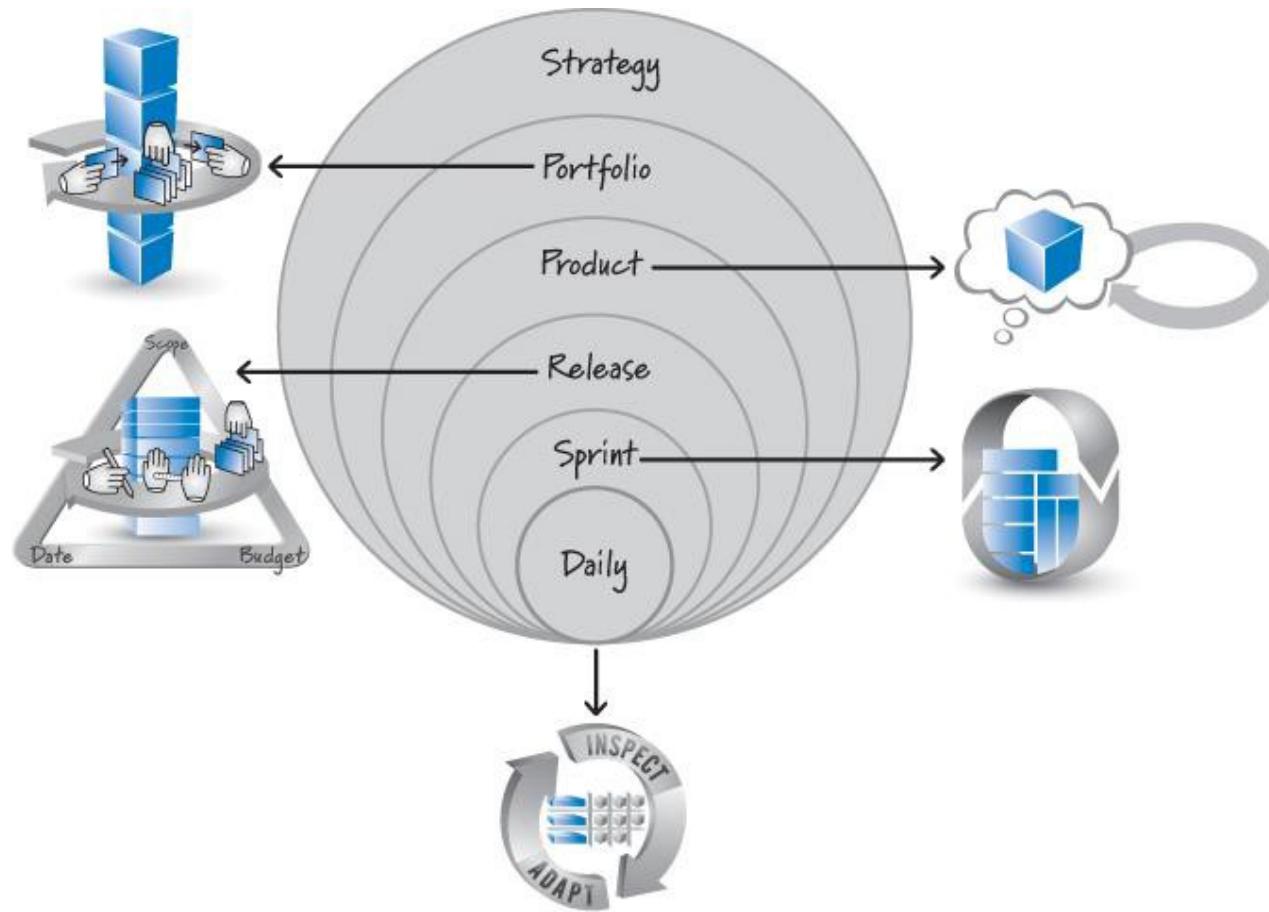
1. Understand the role of a project schedule
2. Understand how to develop a project schedule
3. Understand how to use a project schedule to monitor and track project progress
4. Understand agile planning principles



- 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



Different levels of planning in Scrum

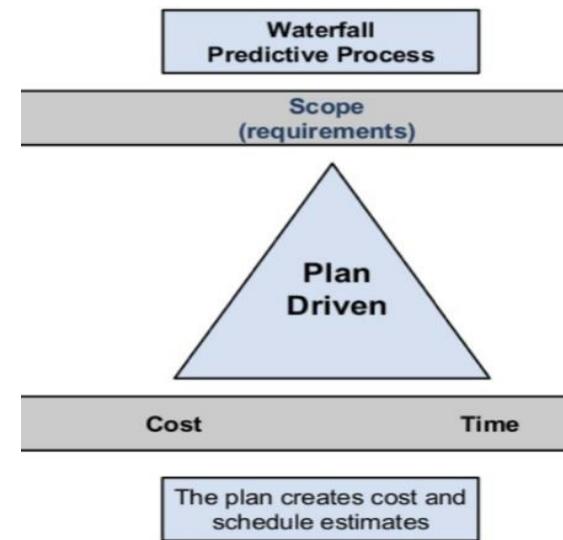


MELBOURNE

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



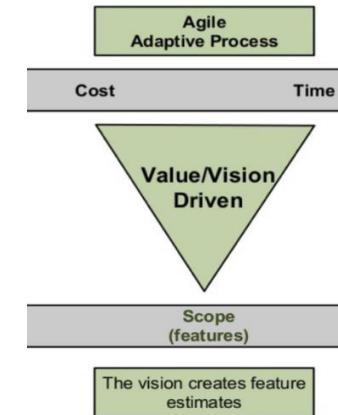
- Assumptions in Formal Planning:
 - Scope fixed – requirements are stable
 - Budget fixed – cost estimations are accurate
 - Schedule fixed - derived based on scope and budget





MELBOURNE

- 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>



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}

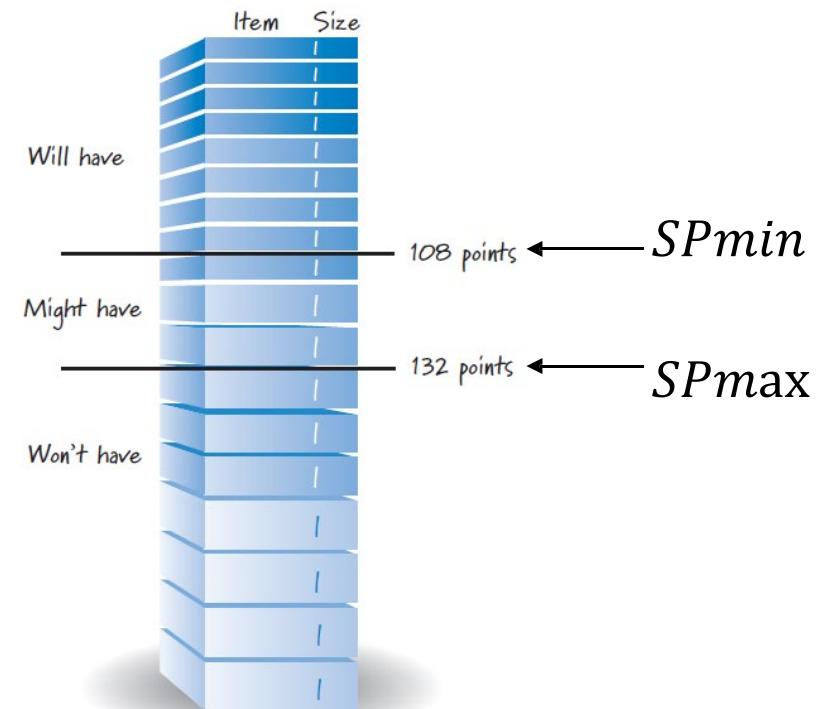
Compute minimum and maximum story points based on velocity

$$SP_{min} = V_{min} \times N,$$

$$SP_{max} = SP_{max} \times N$$

Draw lines through the Product Backlog to show the above

Fixed-Date: used when date is more important





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})

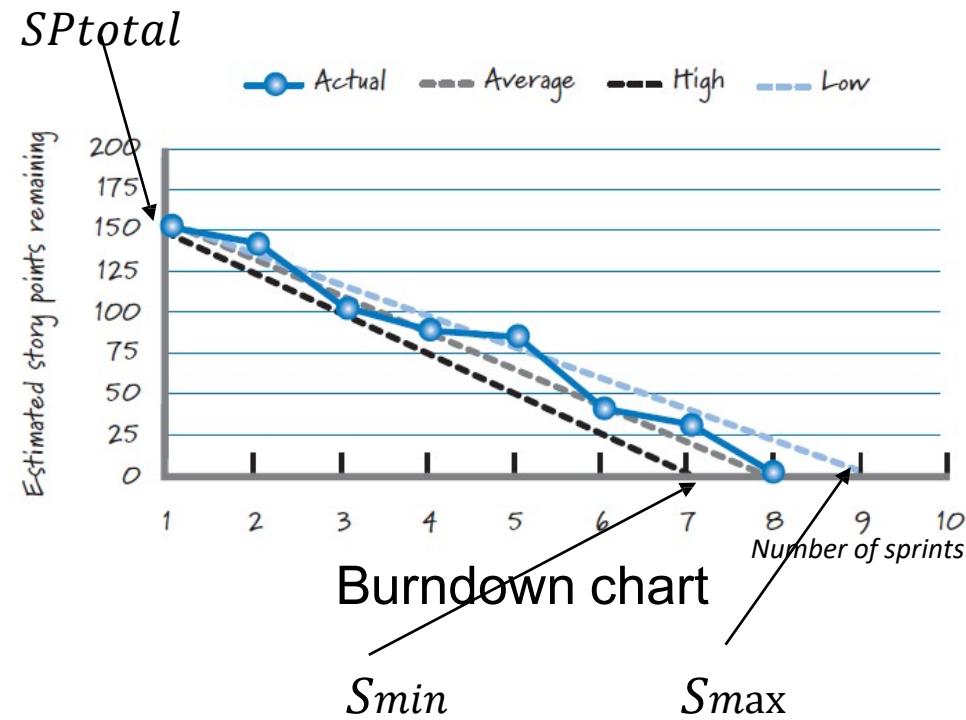
Measure team velocity range:
 V_{min}, V_{max}

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.



SWEN90016
Software Processes & Project Management

Quality Management

Copyright University of Melbourne 2018

Marion Zalk

Department of Computing and Information Systems

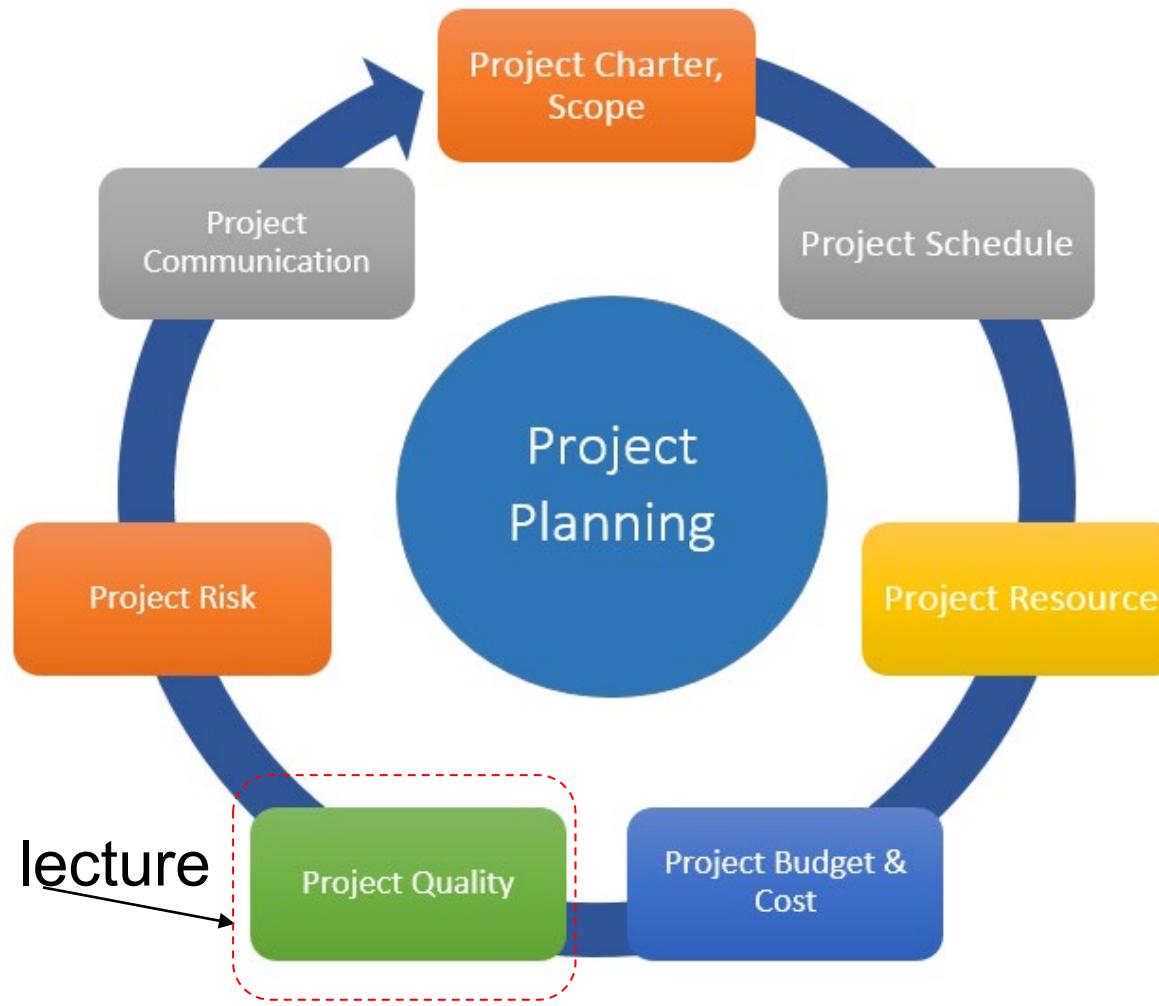
The University of Melbourne

mzalk@unimelb.edu.au

2021 – Semester 1
Lecture 8



WILLING TO LEARN





Quality Management

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



Version 0.1 (2018)

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



Quality is not an act, it is a habit — Aristotle

- Evidence shows that we cannot simply fix up our software *post-hoc* and add in quality attributes after building the system
- Quality must be *built into the software from the beginning*
- In this topic you will learn how to built quality into the software through a range of *Quality Management* activities



- We define quality from two broad perspectives:
 - **End-user's Perspective:**

Typically, end-users judge the quality of a product by their interaction with it. For users, a system has quality if it is fit for purpose, is reliable, has reasonable performance, is easy to learn and use, and helps the users in achieving their goals. Sometimes, if the functionality is hard to learn but is extremely important and worth the trouble of learning, then users will still judge the system to have high quality. These are termed **external quality characteristics**, because they are typically associated with the external behaviour of the system.

- **Developer's Perspective:**

The developer's perspective typically also includes the number of faults that the system has, ease of modifying the system, ease of testing the system, the ease of understanding the system design, the re-usability of components, conformance to requirements, resource usage, and performance. These are mainly **internal quality characteristics**, because they are concerned with the quality of the internal structure of the system.



- Some claim:

Most quality assurance activities are too costly - savings made from not using resources is greater than the cost incurred in fixing the faults

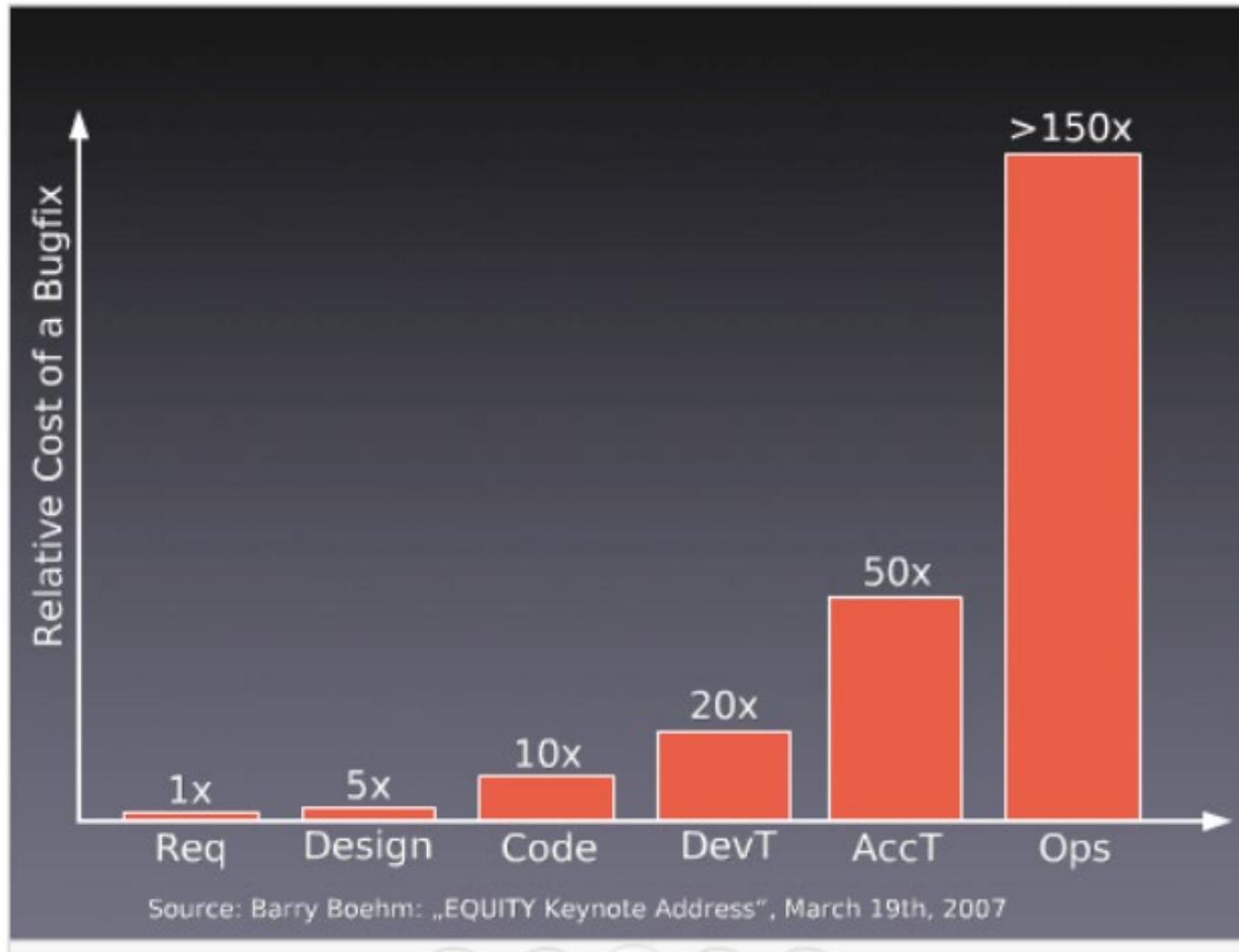
- For example, instead of performing formal reviews of requirements specification documents, it is far better to build the system, ask the client/user for feedback, and to correct any faults from there.
- Alternatively, one can simply release the system and correct faults as users report them.

- Empirical studies refute the above claim:
 - There are many studies in the area



Cost of quality

MILLIONS OF DOLLARS





1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



WILLING TO LEARN





WILLING TO LEARN

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



1. ***Quality assurance:***

The establishment of a framework of organizational procedures and standards that lead to high-quality software

2. ***Quality planning:***

The selection of appropriate procedures and standards from the framework, adopted for the specific project

3. ***Quality control:***

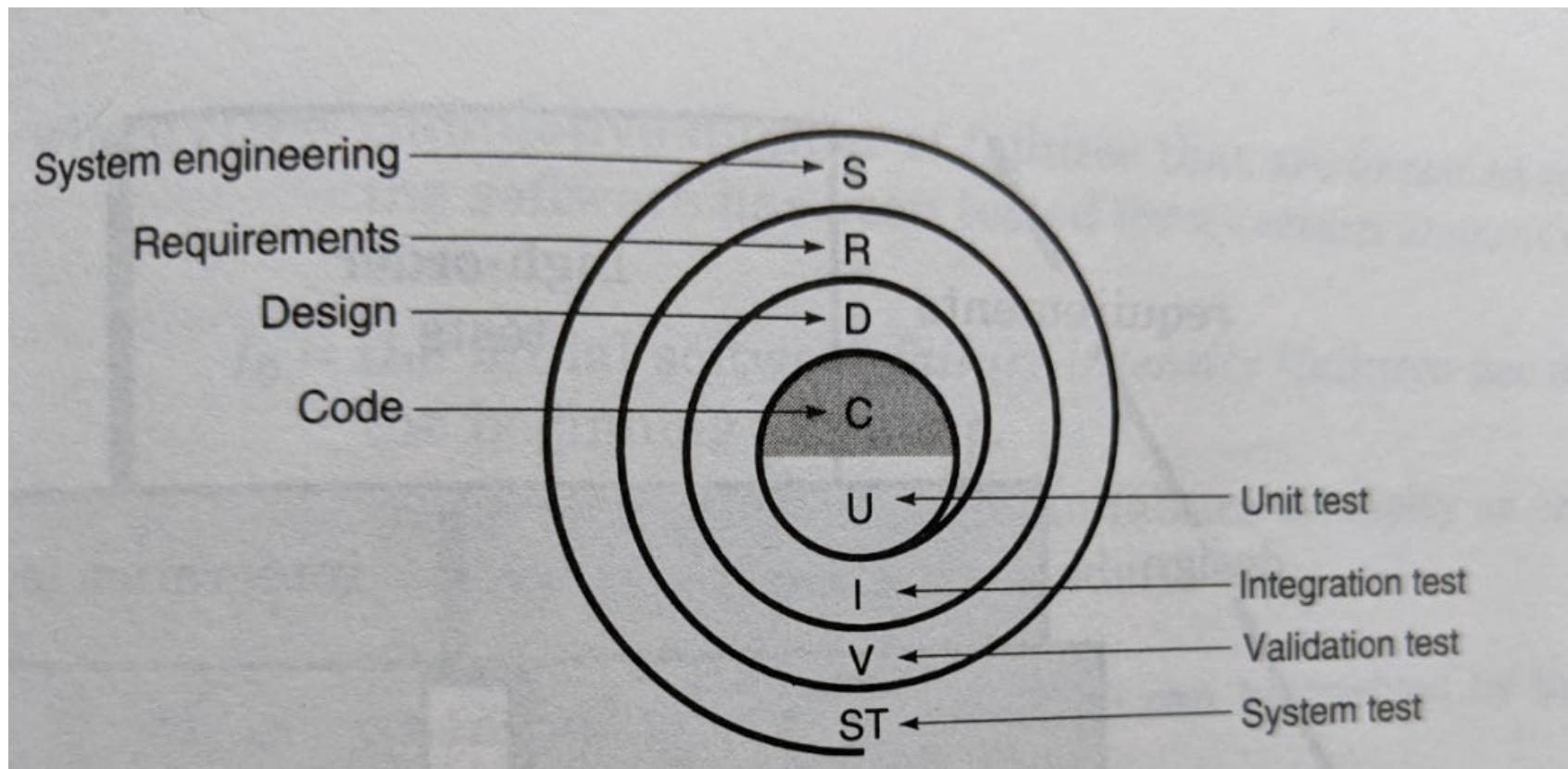
Ensuring that the software development team has followed the project quality procedures and standards



- Verification and Validation (V &V) are important aspects of quality assurance
- **Verification:**
 - Verification is an attempt to ensure that the product is built correctly, in the sense that the output products of an activity meet the specifications imposed on them in previous activities.
 - Verification normally involves two (sets of) artifacts: req. spec. vs design, design vs code; this is an internal developer activity.
 - Verification is ensuring you are *building the system right* (the right way).
- **Validation:**
 - Validation is an attempt to ensure that the right product is built—that is, the product fulfils its specific intended purpose.
 - Validation involves going back to the stakeholders to check if the product meets their requirements; this normally involves something/someone external.
 - Validation is ensuring that you are *building the right system* (to meet stakeholder needs).



WILLING TO LEARN



[1] p421



SW development process

D1

D2

D3

D4

D5

Quality management process

STD's
and
Pro-
cedures

Quality
Plan

Quality review reports



WILLING TO LEARN

- Quality assurance process is primarily concerned with defining or selecting the *quality standards*
 - A standard might simply be defined as a *set of rules for ensuring quality*
 - Standards play an important role in the quality management process
- There are two types of standards:
 - Product standards:
 - These apply to the product being developed
 - Process standards:
 - These standards define the processes that should be followed during software development



WILLING TO LEARN

Product Standards	Process Standards
Design review form template	Design review conduct
Requirements document structure	Design validation process
Documentation standards	Version release process
Coding standards to follow	Project plan approval process
Project plan format	Change control process
Change request form template	Test recording process

Product vs process standards



- Why are documentation standards important?
 - documents are the tangible manifestation of the software
- Documentation process standards
 - How documents should be developed, validated and maintained
- Document standards
 - Concerned with document identification, structure, presentation, changes highlighting, etc.
- Document interchange standards
 - How documents are stored and interchanged between different documentation systems
 - XML is an emerging standard for document interchange which will be widely supported in future



- Advantages of standards
 - Provide a framework around which the quality assurance process may be implemented
 - Provide encapsulation of best, or at least most appropriate, practice
 - Customers sometimes require a particular quality standard/level when choosing a software vendor
- Problems with standards
 - Not seen as relevant and up-to-date by software engineers
 - Involve too much bureaucratic form filling
 - Unsupported by software tools so tedious manual work is involved to maintain standards

Standards should not be avoided, but should be tailored as needed!



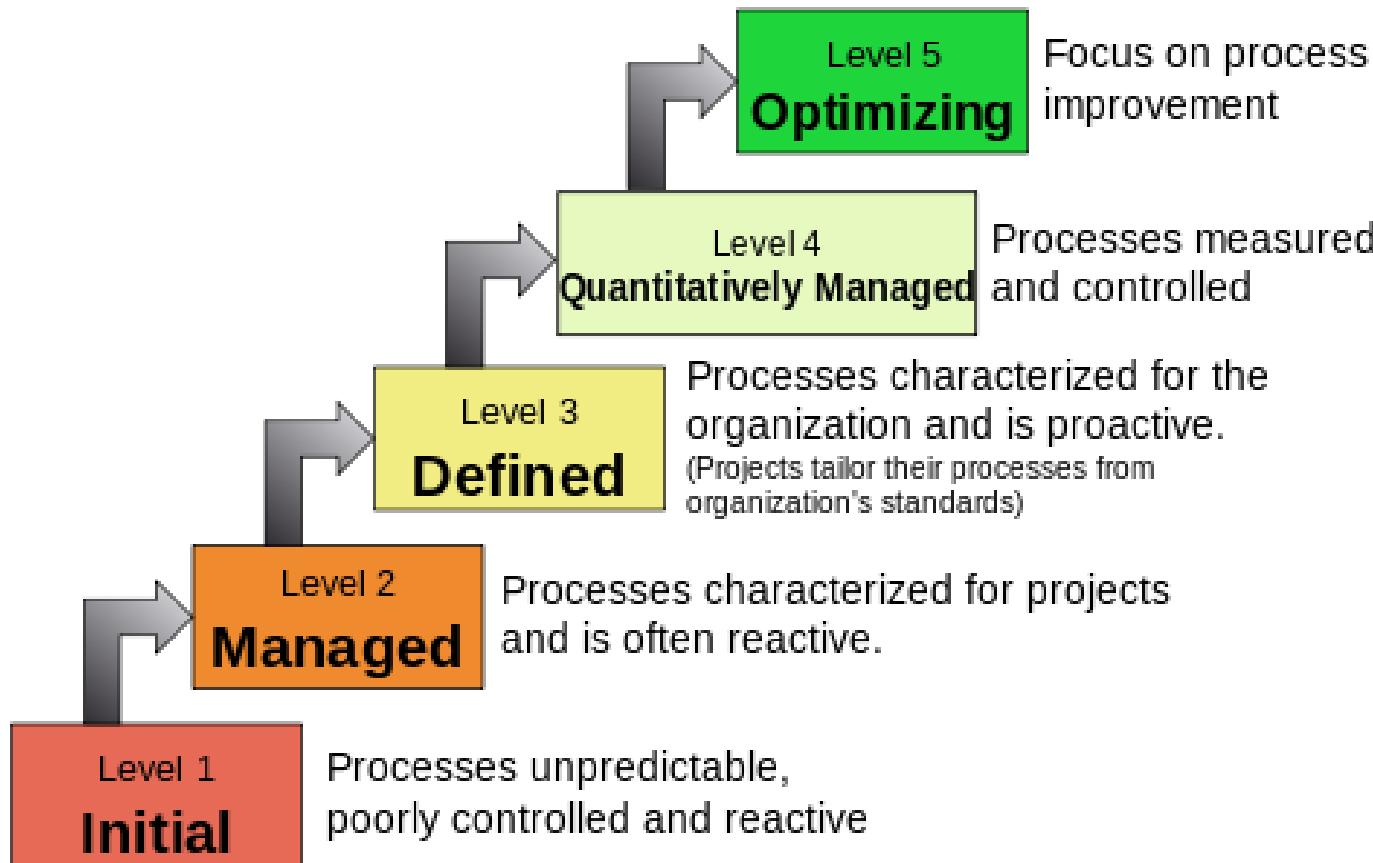
- Many standards and systems related to software quality exists today
- Some examples of software standards and systems
 - ISO 9000
 - Capability Maturity Model



- Developed by the Software Engineering Institute (SEI) at Carnegie Mellon University
- Describes the key elements of an effective software development process
- Describes an approach for software companies to move from an ad-hoc, immature process to a mature developed process
- Organizations are characterised being at a Level from 1-5 based on the processes they follow

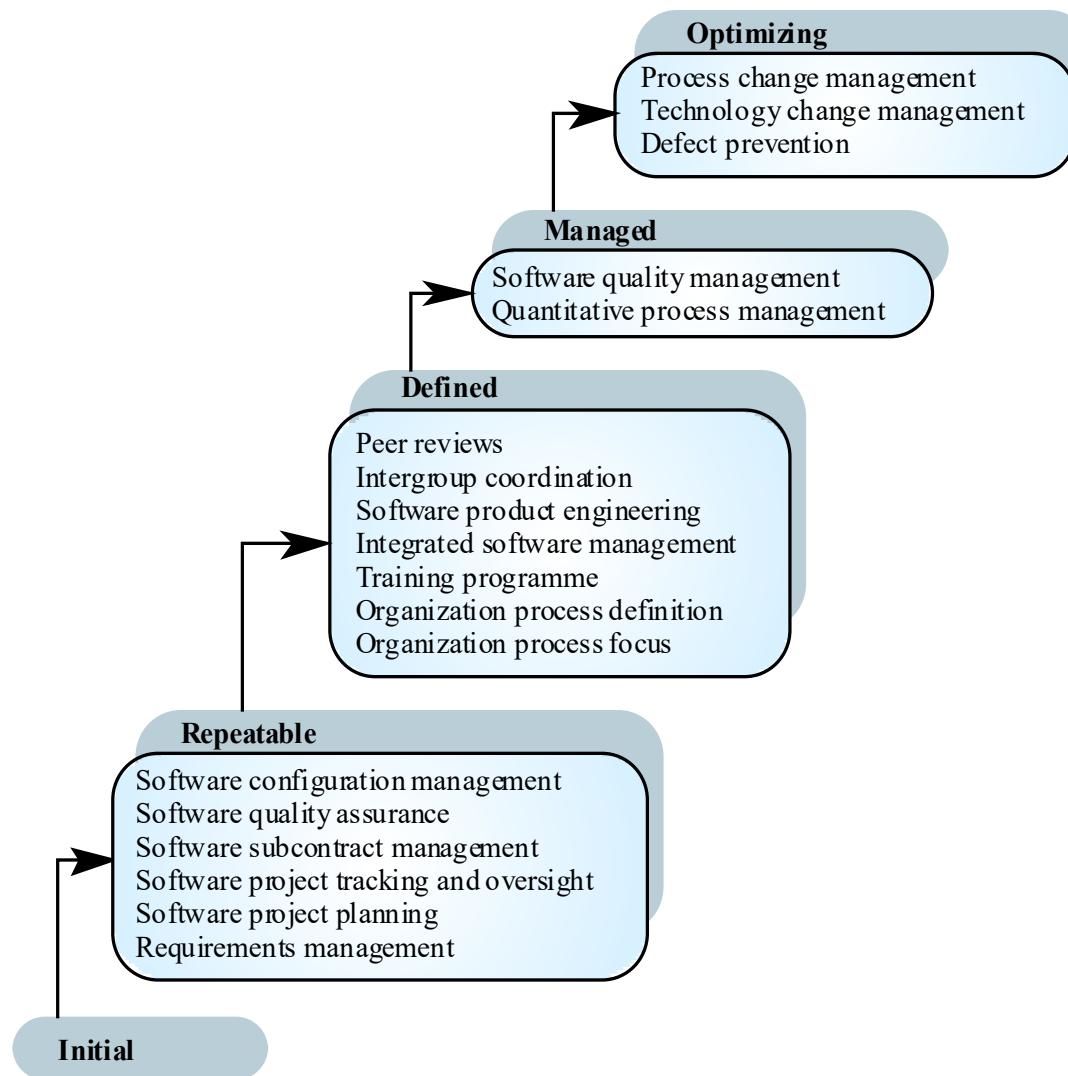


Characteristics of the Maturity levels





Version 2.0





WILLING TO LEARN

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



1. *Quality assurance:*

The establishment of a framework of organizational procedures and standards that lead to high-quality software

2. *Quality planning:*

The selection of appropriate procedures and standards from the framework, adopted for the specific project

3. *Quality control:*

Ensuring that the software development team has followed the project quality procedures and standards



- The process of selecting those standards and systems that are appropriate to a particular organization and project
- The outcome of the planning process is a:
 - Software Quality Plan (SQP), sometimes called a Software Quality Assurance Plan (SQAP)



- **Software Quality Assurance Plan**
 - Product Overview

A description of the product, intended market, and quality expectations
 - Product Plan

The critical release dates and responsibilities – could point to the schedule
 - Quality Goals

The quality goals and plans for the product, including identification and justification of critical product quality attributes
 - Process Description

The quality assurance processes that should be used for product development and management (reviews, audits etc)
 - Document and Coding Standards

Standards for the documents and coding standards
 - Risks and Risk Management

The key risks that might affect product quality and the actions to address these risks (could provide a link to appropriate risks in the Risk Management Plan)



SWEN90016

Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Software quality attributes

- Some of the quality attributes matter only to developers while others matter to end-users
- It is not possible for any system to be optimised for all attributes – trade-off is necessary to select the most important ones

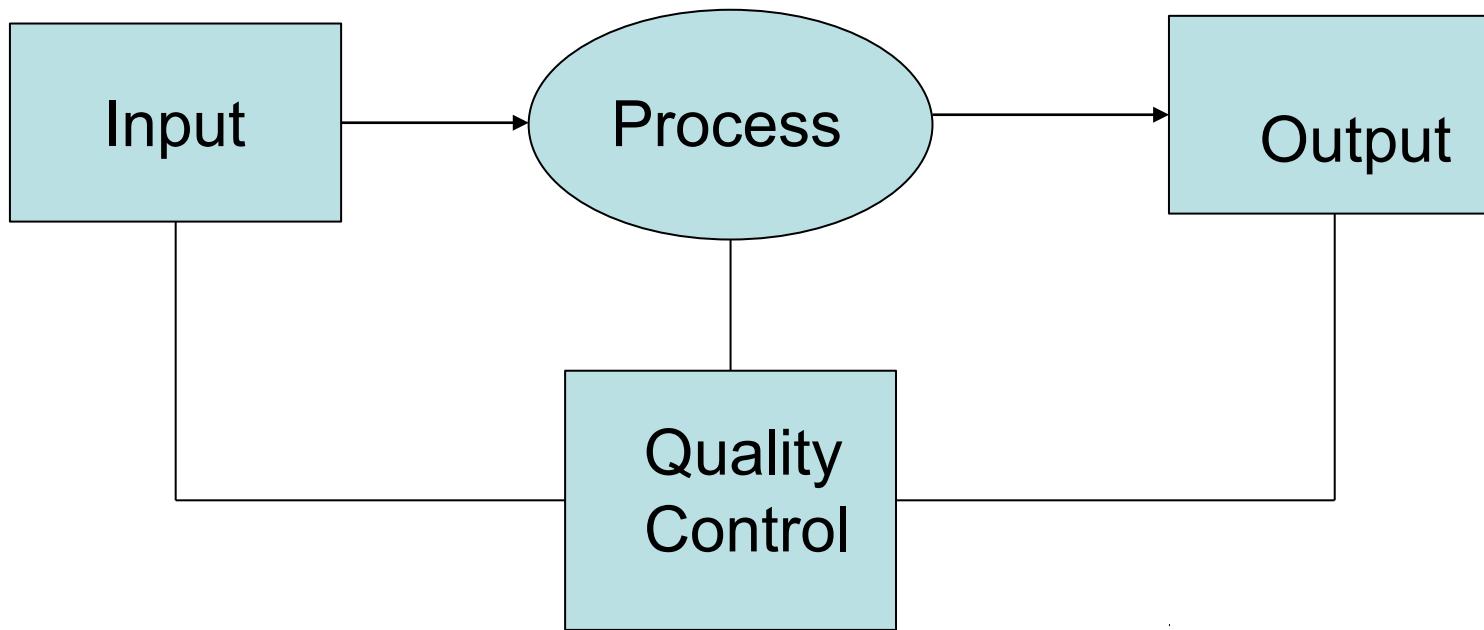


WILLING TO LEARN

1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



- Involves monitoring the software development process to ensure that the quality assurance procedures and standards specified in the SQP are being followed





- **Review** is a common technique used for verification and validation
- Artefacts produced during the development process are reviewed as a way of identifying problems seeking ways to improve them early
- Three common types of reviews:
 - Technical Reviews
 - Business Reviews
 - Management Reviews



- Reviews of artefacts is performed by **peers** in the development team but the author/s are involved
- The aim is uncovering problems in an artefact and seeking ways to improve the artefact
- Is considered a “soft” method for quality assurance - that is, nothing is executed
 - Some developers greet reviews with scepticism - however, empirical evidence suggests that such scepticism is unjustified



- Advantages of technical reviews:

- **Can be performed on any software artefact**, whereas many “hard” methods of quality assurance, such as testing and measurement, can be performed only on executable artefacts.
- **Earlier detection of problems** in software artefacts leads to lower costs of resolution.
- Studies show that roughly 30-70% of all programming faults found in a project were located using **source code reviews**, and up to 80% according to studies performed by IBM. Some studies demonstrated that **review techniques found several types of faults that testing failed to find**, and vice-versa.
- **Reviews find the actual faults in source code**, in contrast to testing, which merely indicates that there is a fault somewhere in the program. After a fault is detected with testing, it must then be located.
- Due to internal pressure of getting software releases out the door, programmers make more mistakes when correcting faults that were found during testing than they do **correcting faults during the review phase**



- Disadvantages of technical reviews:
 - Could be time and resource consuming
 - Should be carefully planned and executed to get the desired outcomes
- Types of technical reviews
 - Informal Reviews
 - Formal Reviews
 - Walk throughs
 - Code inspections
 - Audits



- **Informal Reviews:**

- A simple desk check or casual meeting with a colleague which aims to improve the quality of a document
- No formal guidelines or procedures that are followed
- The effectiveness of informal reviews is considerably less than formal reviews, because of the lack of diversity found in a group
- Checklists are tools that can help to improve the effectiveness of a review.
- A checklist is a list of questions that the reviewer must answer about an artefact, however, the questions are generic questions about that type of artefact
- Less time and resource consuming than a formal review



Example checklist for a Requirements Specification

Checklist for software requirements specification artifact	
Organisation and Completeness	
<input type="checkbox"/> Are all internal cross-references to other requirements correct?	
<input type="checkbox"/> Are all requirements written at a consistent and appropriate level of detail?	
<input type="checkbox"/> Do the requirements provide an adequate basis for design?	
<input type="checkbox"/> Is the implementation priority of each requirement included?	
<input type="checkbox"/> Are all external hardware, software, and communication interfaces defined?	
<input type="checkbox"/> Have algorithms intrinsic to the functional requirements been defined?	
<input type="checkbox"/> Does the specification include all of the known customer or system needs?	
<input type="checkbox"/> Is the expected behaviour documented for all anticipated error conditions?	
Correctness	
<input type="checkbox"/> Do any requirements conflict with or duplicate other requirements?	
<input type="checkbox"/> Is each requirement written in clear, concise, unambiguous language?	
<input type="checkbox"/> Is each requirement verifiable by testing, demonstration, review, or analysis?	
<input type="checkbox"/> Is each requirement in scope for the project?	
<input type="checkbox"/> Is each requirement free from content and grammatical errors?	
<input type="checkbox"/> Is any necessary information missing from a requirement? If so, is it identified as "to be decided"?	
<input type="checkbox"/> Can all of the requirements be implemented within known constraints?	
<input type="checkbox"/> Are any specified error messages unique and meaningful?	
Quality Attributes	
<input type="checkbox"/> Are all performance objectives properly specified?	
<input type="checkbox"/> Are all security and safety considerations properly specified?	
<input type="checkbox"/> Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?	
Traceability	
<input type="checkbox"/> Is each requirement uniquely and correctly identified?	
<input type="checkbox"/> Is each software functional requirement traceable to a higher-level requirement (e.g., system requirement, use case)?	
Special Issues	
<input type="checkbox"/> Are all requirements actually requirements, not design or implementation solutions?	
<input type="checkbox"/> Are all time-critical functions identified, and timing criteria specified for them?	
<input type="checkbox"/> Have internationalisation issues been adequately addressed?	



• **Formal Reviews:**

- A meeting with multiple stakeholders such as developers, testers, client
 - The group approach has benefits of bringing out different perspectives
- Meeting should adhere to the following constraints
 - The review team should be 3-5 members carefully chosen
 - The meeting should last no longer than 90 minutes
 - Following are the critical roles
 - Review Leader: responsible for organizing the review
 - Author: at least one author should be present
 - Reviewers: at least two or three non-author stakeholders
 - Recorder: responsible for recording all important review comments
- The review meeting could recommend one of the following:
 - Accept without further changes
 - Accept with proposed changes
 - Reject the artefact – this requires a re-review after modifications





- **Walkthroughs**

- Walkthrough could be for code or a document
- This is a review process where the author (the programmer or designer) leads a group of reviewers
- Following are the main differences from a formal review:
 - Moderator, that leads the review is the author of the artefact being reviewed
 - Reviewers do not need preparation
 - When defects or inconsistencies are found, possible solutions are discussed

- **Code Inspections**

- These are very similar to formal reviews, expect that the focus is on the code



- **Audits**

- Reviews of processes and products to determine if a particular product or process conforms to standards
- It is a type of technical review where the authors of the artefact being audited are not involved in the audit process at all – all the other roles are similar to a formal review
- Audits are typically performed by a team that is completely external to an organisation
- Two types of audits:
 - Product audits: to confirm that the product meets the standards
 - Process audits: to ensure that the team follows processes



- The goal of a business review is to ensure that the IT solution provides the functionality specified in the project scope and requirements document
- A business review can include all project deliverables to ensure that:
 - It is complete
 - Provides the information needed to move to the next phase or process
 - Meets the standards

WILLING TO LEARN

- Compares the project's actual progress against a baseline project plan
- Project Manager is responsible for presenting the project progress and providing a clear picture of the current status
- Issues need to be resolved – e.g. resources reallocated as needed, change to the project course if needed
- May involve reviewing if the project meets the scope, schedule, budget and quality objectives



1. Understand the fundamentals of quality management
2. Understand the quality management process
3. Understand the following quality management activities:
 - Quality Assurance
 - Quality Planning
 - Quality Control and Monitoring



1. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
2. I Somerville. Software Engineering, Addison-Wesley Publishing, ninth edition, 2010.
3. ISO. Information technology – software product evaluation – quality characteristics and guidelines for their use, international organization for standardization. International Standard ISO/IEC 9126, International Electrotechnical Commission, Geneva, 1991.



4. Marco Palomino, Abraham Dávil, Karin Melendez, Marcelo Pessoa. Agile Practices Adoption in CMMI Organizations: A Systematic Literature Review. International Conference on Software Process Improvement, 2016.

Configuration Management

Copyright University of Melbourne 2018

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

2021 – Semester 1
Lecture 10



1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management

1. Understand the role of configuration management

<https://www.itnews.com.au/news/coles-shuts-supermarkets-due-to-payment-system-outage-554503>

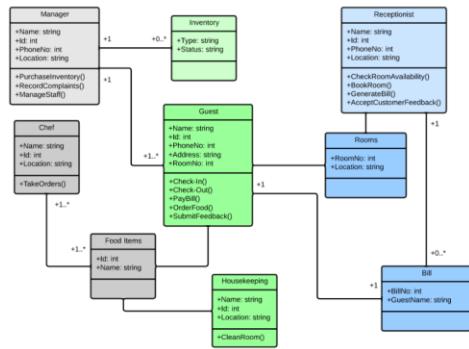
<https://www.abc.net.au/news/2020-10-09/coles-experience-nationwide-closure-over-it-outage/12749358>



- Software projects generate a large number of different types of artefacts – e.g.:



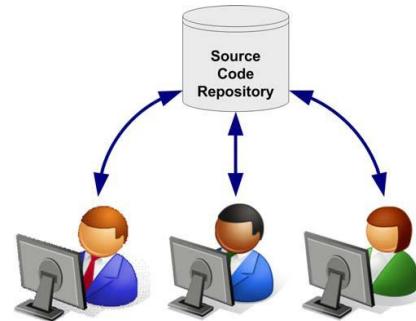
Use-case diagrams



Class diagrams

ITEM	DESCRIPTION
Title	A unique and descriptive title for the test case
Priority	The relative importance of the test case (critical, nice-to-have, etc.)
Status	For live systems, an indicator of the state of the test case. Typical states could include: Design – test case is still being designed Ready – test case is complete, ready to run Running – test case is being executed Pass – test case passed successfully Failed – test case failed Error – test case is in error and needs to be rewritten
Initial configuration	The state of the program before the actions in the "Steps" are to be followed. All too often this is omitted and the reader must guess or intuit the correct pre-requisites for conducting the test case.
Software Configuration	The software configuration for which this test is valid. It could include the version and release of software-under-test as well as any relevant hardware or software platform details (e.g. WinXP vs Win95)
Steps	An ordered series of steps to conduct during the test case, these must be detailed and specific. How detailed depends on the level of scripting required and the experience of the tester involved.
Expected behaviour	What was expected of the software, upon completion of the steps? What is expected of the software. Allows the test case to be validated with out recourse to the tester who wrote it.

Test Cases



Source Code

Software Requirements Specification

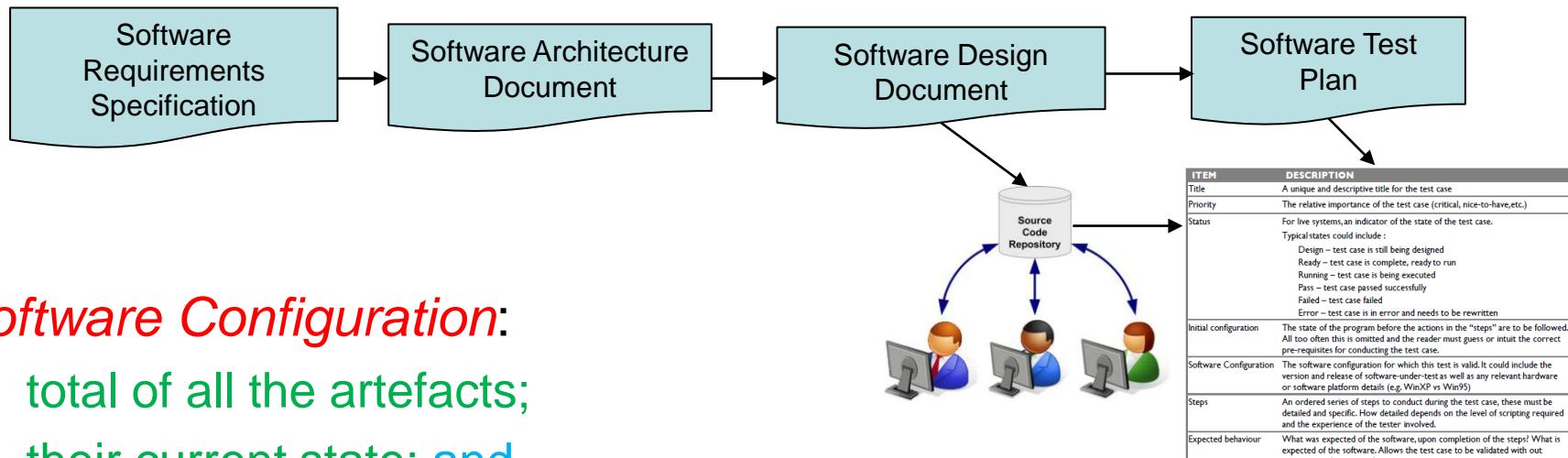
Software Architecture Document

Software Design Document

Software Test Plan



- There are *dependencies* between all of these artefacts.
 - For example, a code module may depend on a design element such as a class diagram or state chart, as well as on a design element such as a design class diagram. In turn these may depend on a combination of textual requirements, use-cases and analysis classes.



- *Software Configuration*:
 - total of all the artefacts;
 - their current state; and
 - the dependencies between them.



The problem is change!

- If we make a change to an artefact, it may impact all of that artefact's dependencies
- If we are not careful then changes to artefacts may leave the configuration in an inconsistent state
 - For example, a change to the requirement will have an impact on the system design and all of the code modules that depend on the design. Also, the test plan, test cases and testing scripts for the code will also be impacted. *The danger is that we may change one module without changing one of its dependent modules leaving the configuration inconsistent.*



- The aim of configuration management is to manage change properly without losing overall consistency through:
 - establishing processes;
 - setting up repositories; and
 - using other appropriate tools and techniques
- Configuration Management (CM) addresses the following:
 - How do we manage requests for change?
 - What and where are the software components?
 - What is the status of each software component?
 - How does a change to one component affect others?
 - How do we resolve conflicting changes?
 - How do we maintain multiple versions?
 - How do we keep the system up to date?

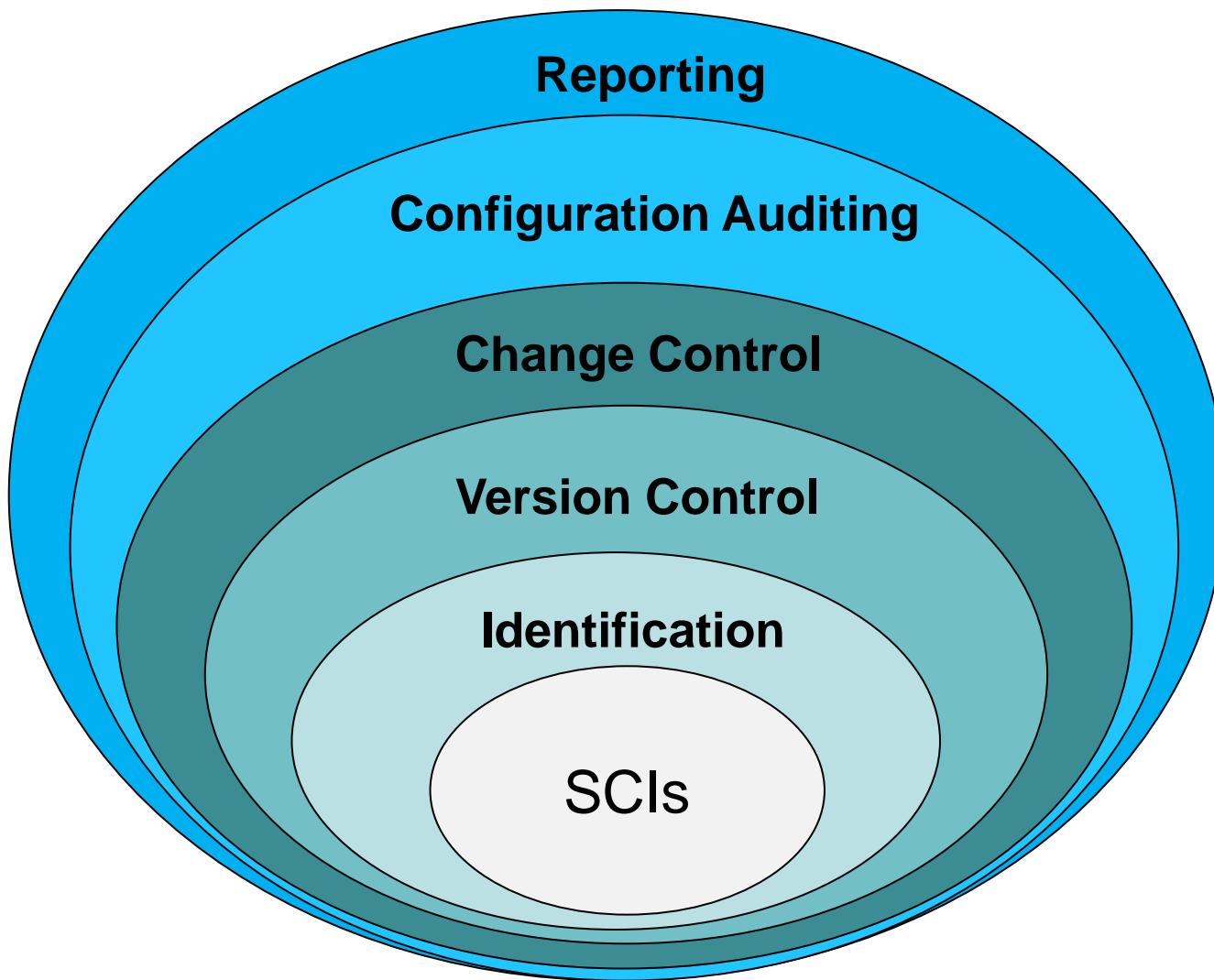


1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management



CM Aims:

1. To identify all items that collectively will make up the configuration
2. To manage changes to one or more of these items so that the collection remains consistent
3. To manage different versions of the product
4. To assure software quality as the configuration evolves over time





- Identification
 - the configuration items necessary for the project are identified
- Version control
 - processes and tools are chosen to manage the different versions of configuration items as they are developed
- Change control
 - changes that affect more than just one configuration item are managed
- Configuration auditing
 - the consistency of the configuration is checked
- Configuration reporting
 - the status of configuration items is reported

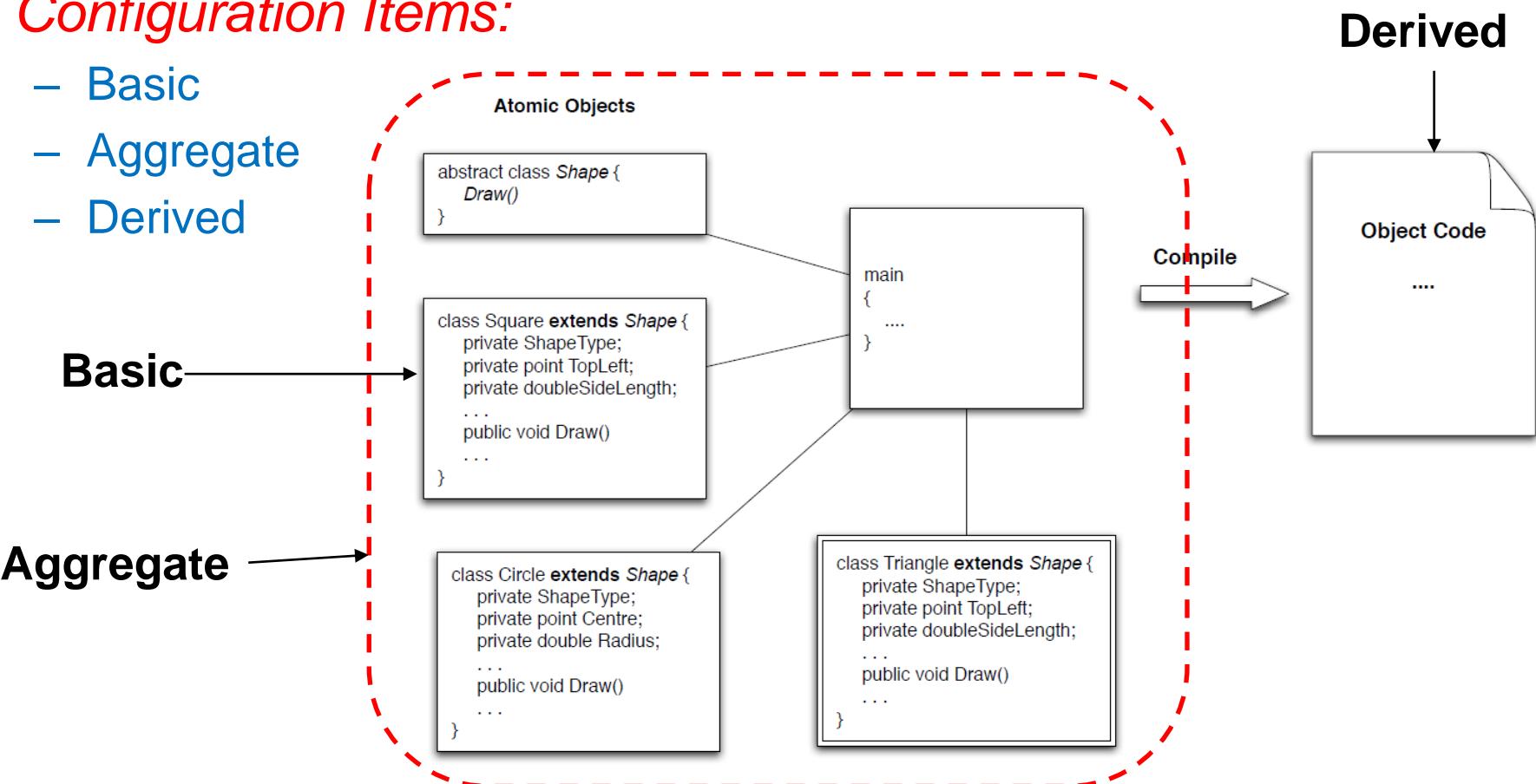


1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management



- The set of artefacts that require configuration management are called the *configuration items*
- Configuration Items:*

- Basic
- Aggregate
- Derived





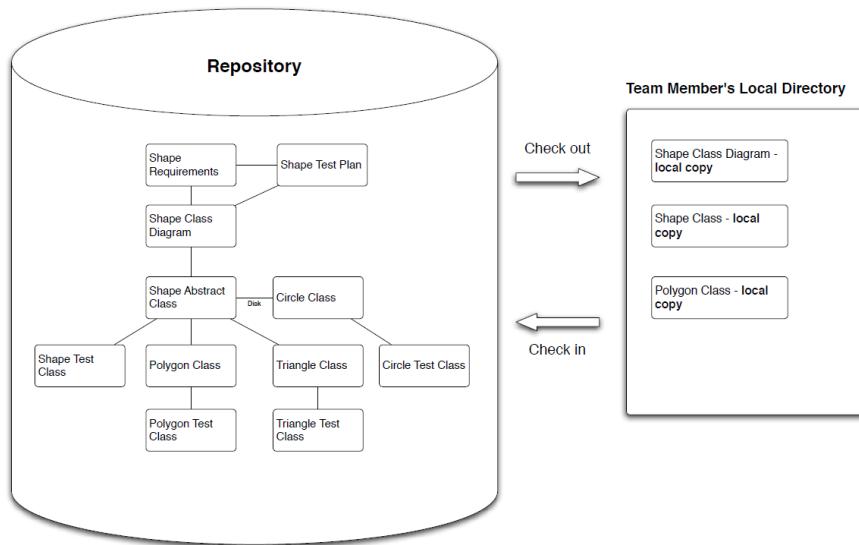
- A typical list of configuration items
 - requirements specifications, requirements models, sections of the requirements specification, and individual requirements
 - use-cases, user stories
 - design models, design documents, design elements, and class designs
 - source code modules
 - object code modules
 - release modules
 - software tools
 - test drivers and stubs, and test scripts
 - documents or sections of documents associated with the project



- Requirements for a version control system:
 1. A *repository* for storing configuration items
 2. A *version management function* that allow software engineers to create and track versions, and roll the system back to previous versions if necessary – e.g. git, svn, cvs
 3. A *make-like facility* that allows engineers to collect all of the configuration objects for a particular target together and to build that target – e.g. *Apache Maven*, *Apache Ant*, *make (unix, linux)*



- SCM information is maintained in a *repository* or *configuration database*



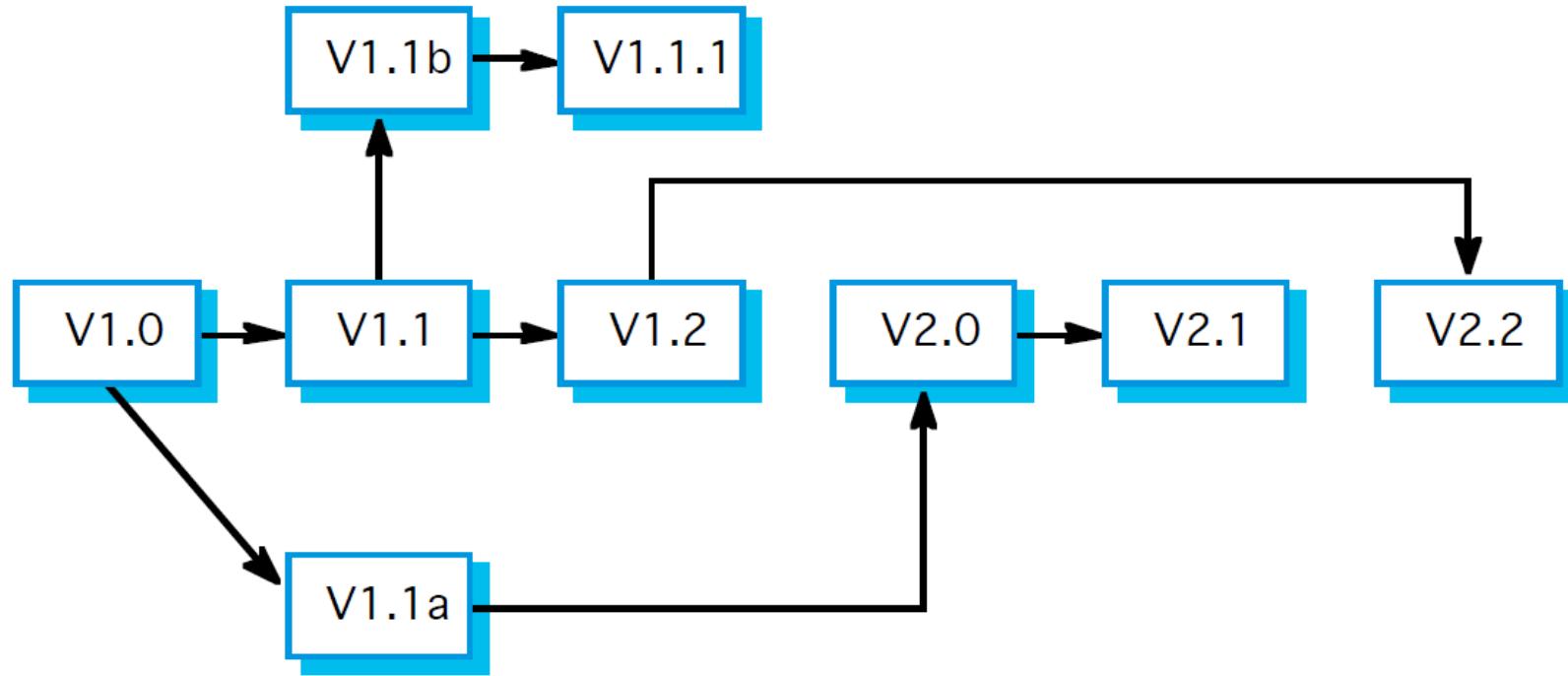
Version: An instance of a model, document, code, or other configuration item which is functionally distinct in some way from other system instances.

Variant: An instance of a system which is functionally identical but non-functionally distinct from other instances of a system.

Release: An instance of a system which is distributed to users outside of the development team.



- *Derivation History:*
 - This is a record of changes applied to a configuration object
- Each change should record:
 - the change made
 - the rationale for the change
 - who made the change
 - when it was implemented
- A common method of tracking versions in a repository is through *version numbering*
 - Version numbers could have meanings – for example a reviewed version of a document (major versions) vs un-reviewed changes



A derivation structure for a project using version numbering to mark branches and merges



- Version control
 - processes and tools are chosen to manage the different versions of configuration items as they are developed
 - A project database that stores all relevant configuration objects,
 - A version management capability that stores all version of configuration object,
 - A make facility that enables the software engineer to collect all relevant configuration objects,
 - Construct a specific version of the software.
- Change control
 - changes that affect more than just one configuration item are managed
 - Change control is manual step in software lifecycle. It combines human procedures and automated tools.
 - Change request submitted and evaluated to assess technical merit, potential side effects, overall impact on other configuration object and system function, and project cost of change.



- *Change Management Plan*
 - A part of an overall configuration management plan to specifically control these changes to the configuration
 - Changes must be made in a way that allows everyone on the project team to find out:
 - exactly what changes need to be made
 - what they need to do to affect the change
 - why the change is being made
 - how it will impact them

More importantly, in distributed control structures, some changes may need to be carefully negotiated so that everyone understands the need for the change and supports it



Element	Impact on the Process
Initiate the Change	Why is the change being made? What information will be needed to evaluate the change? How will the change be evaluated?
Evaluate the Change	How will the change affect the configuration? Which artefacts need to change and what are their dependencies? What are the benefits of the change? What are the costs of the change? Do the benefits of the change outweigh the costs of the change? Who will be impacted by the change?
Making the Change	Who will put the change into effect? How will the change be managed? How will other people working on the project understand the change? How will they be notified of the change? How will people working on the project know when the change is completed?



- ***Baseline***
 - A baseline is an artefact that is *stable*
 - It has been formally reviewed and agreed upon, that is now ready for future development
 - It can only be changed through a *formal change management procedure*

- *Configuration audits:*
 - complement the other configuration management activities by assuring that what is in the repository is actually consistent and that all of the changes have been made properly

Have the changes requested and approved been made?	Have any additional changes other than required by a request been made?
Did the configuration objects that were changed pass their quality assurance tests?	Do the objects in the configuration meet the required external standards?
Do the attributes of the configuration item match the change?	Does every configuration item have appropriate change logs?

Typical questions for a configuration audit



- *Status Reporting*
 - Is a common way for large projects to keep track of the status of the repository
 - The idea is to review the configuration objects for consistency with other configuration objects, to find any omissions or to look for potential side effects
 - Status reporting can take many forms, but most commonly the aim is to report on the status of the configuration items of interest and the baselines that have been achieved
 - For example, we may have a design element that is in one of the states: not-initiated, initial-work, modified, approved, baselined – the status report can compare the state with what is in the project schedule



1. Understand the role of configuration management
2. Understand the configuration management process
3. Understand the tasks associated with configuration management



1. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
2. I Somerville. Software Engineering, Addison-Wesley Publishing, ninth edition, 2010.
3. ISO. Information technology – software product evaluation – quality characteristics and guidelines for their use, international organization for standardization. International Standard ISO/IEC 9126, International Electrotechnical Commission, Geneva, 1991.



4. Marco Palomino, Abraham Dávil, Karin Melendez, Marcelo Pessoa. Agile Practices Adoption in CMMI Organizations: A Systematic Literature Review. International Conference on Software Process Improvement, 2016.



SWEN90016
Software Processes & Project Management

Cost Estimation

Marion Zalk

Department of Computing and Information Systems

The University of Melbourne

mzalk@unimelb.edu.au

2021– Semester 1
Lecture 7

Copyright University of Melbourne 2018

Assignment 2

MATERIALS FOR THIS

Your Assignment 2 report should include screen shots of your Gantt Chart or Kanban board in section 6.5.

You have an option of also including a link to your online tool, such as Trello, if you think this will aid your marker, as sometimes the screen shots are fuzzy and difficult to read.

If you grant (read) permissions to the generic SWEN90016 tutor email address inside your tool,

swen90016tutors@groups.unimelb.edu.au

then all tutors who click on the link inside your report will have access to read.



WILLING TO LEARN



|



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 Scheduling

Redecorate Room

Prepare materials

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

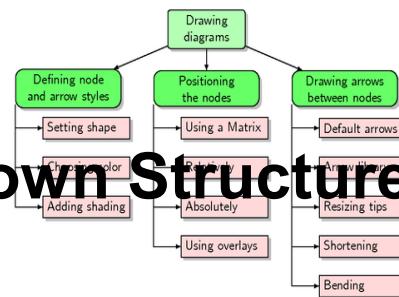
Prepare room

- Remove old wallpaper
- Remove tacks from the floor
- Cover floor with old newspapers
- Cover electrical outlets/switches with tape
- Cover furniture with sheets

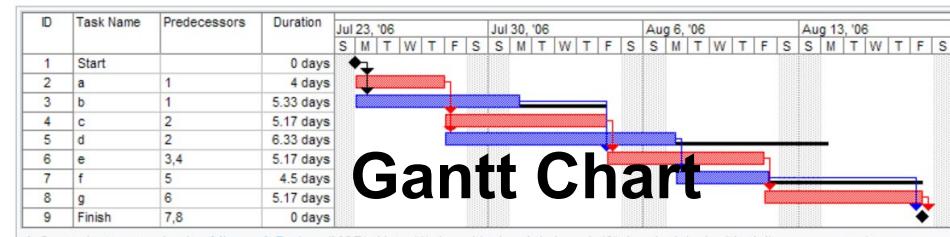
Paint the room

Clean up the room

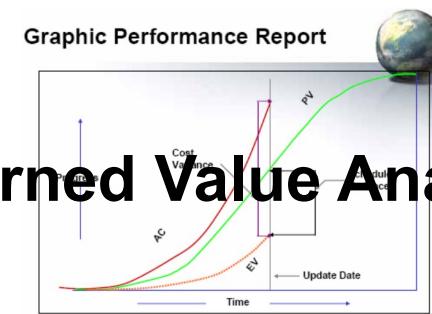
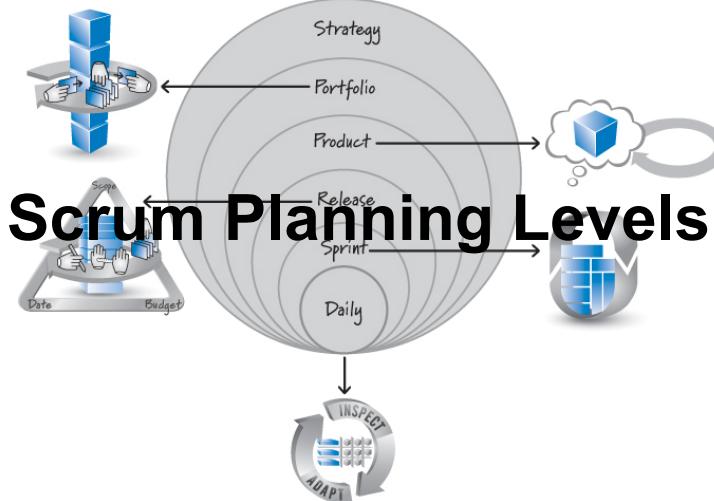
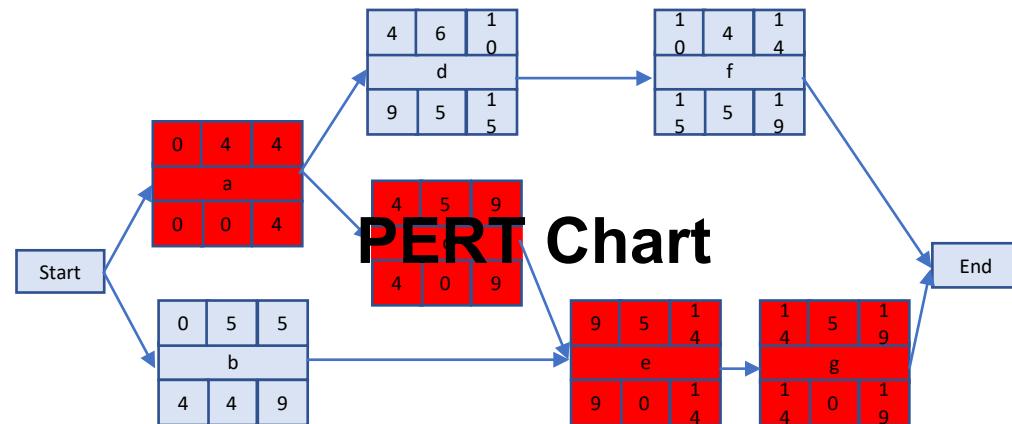
- Dispose or store leftover paint
- Clean brushes/rollers
- Dispose of old newspapers
- Remove covers



Breakdown Structure



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.





WILLING TO LEARN



Today's lecture



WILLING TO LEARN

Formal

1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation

Agile

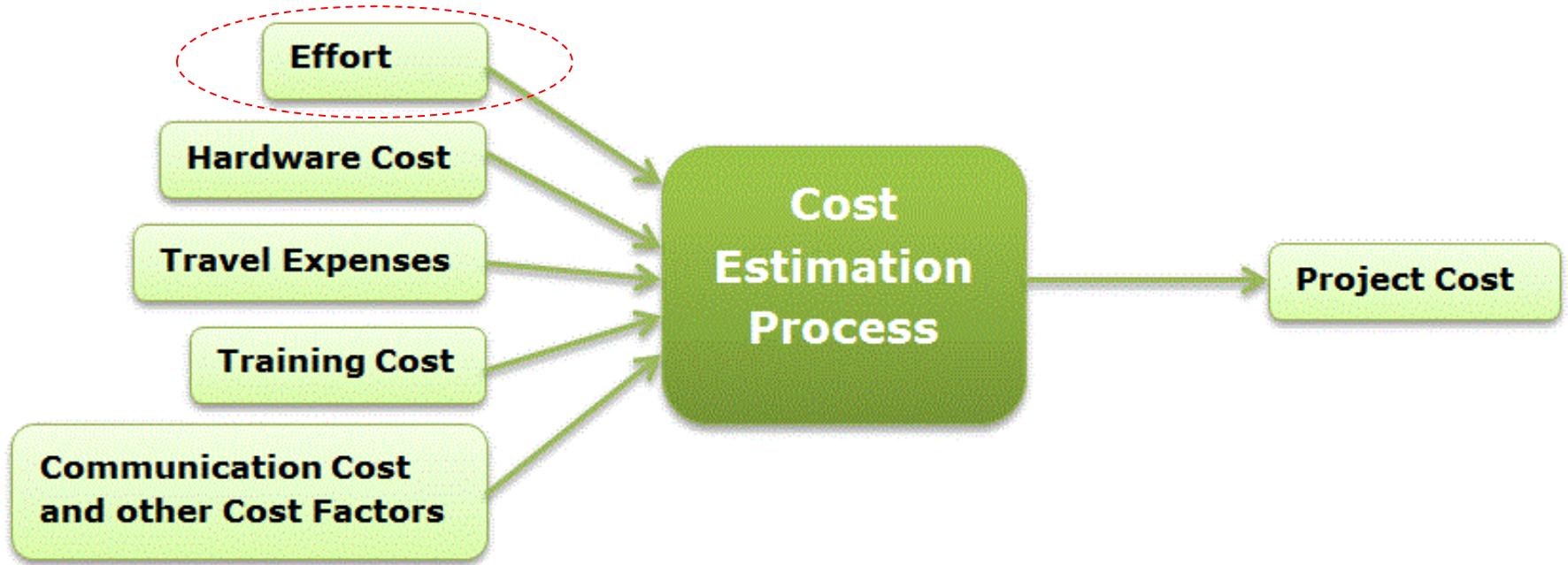
Understand cost estimation techniques used in Agile software development lifecycles



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation



WILLING TO LEARN



<http://geethanjalisnsce.blogspot.com.au/2015/04/>



WILLING TO LEARN

- What is estimation?
 - Is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if *input data may be incomplete, uncertain, or unstable*
- What is *software cost estimation*?
 - Estimation of how much *money, effort, resources, and time* will take to build a specific software based system or product
- Why is it Important?
 - Would you build a house without knowing how much you were about to spend - of course not
 - Since most software systems cost considerably more to build than a large house, it would seem reasonable to develop an estimate before you start creating the software



- There is no exact science for cost estimation – it will never be considered as all accurate
- No person can reasonably predict what can go wrong in the project
- Most estimation methods assume things will proceed as expected and simply adds some slack to account for what can go wrong

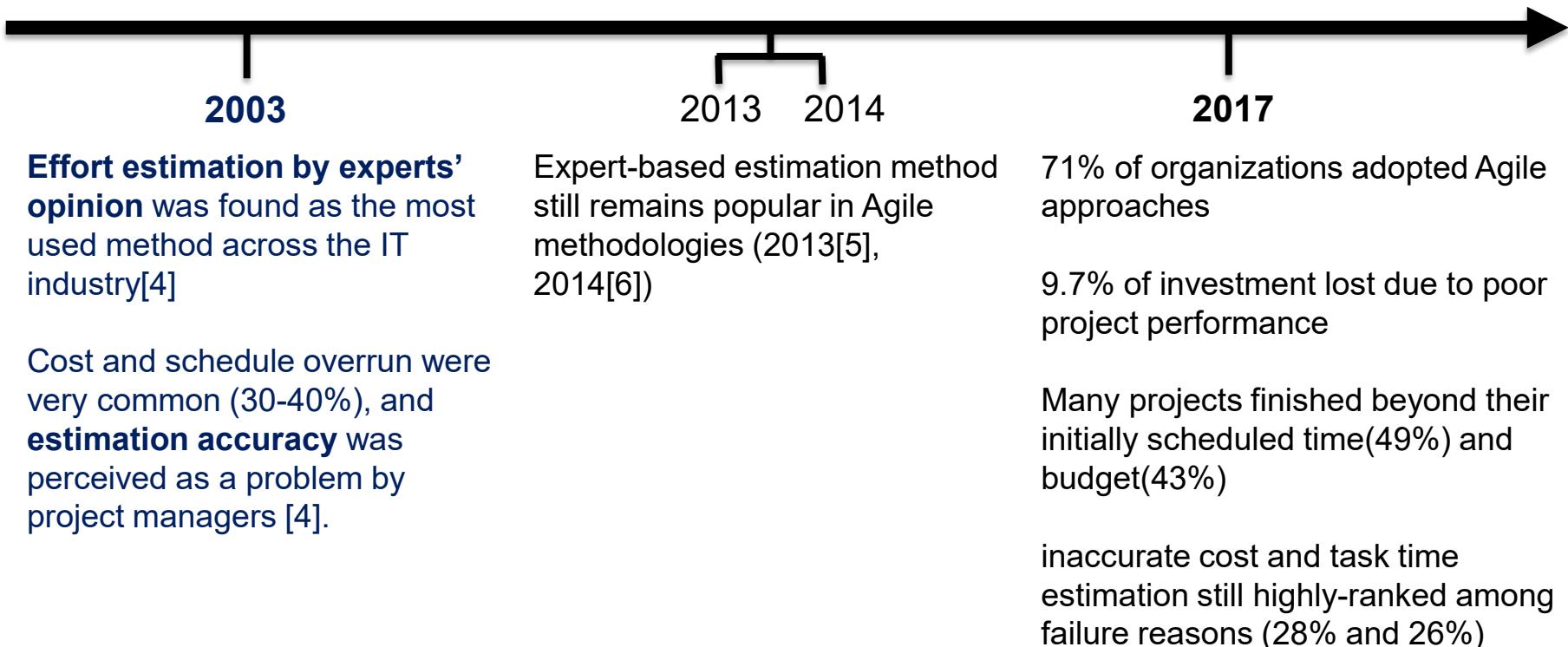


1. Delay estimation – 100% accuracy at the end of the project but less useful!
2. Base estimation on data from previous projects that have been completed
3. Break the system to smaller parts and generate the estimates for smaller parts, which is easier
4. Use empirically-based estimation methods



MILLION OF FIRM

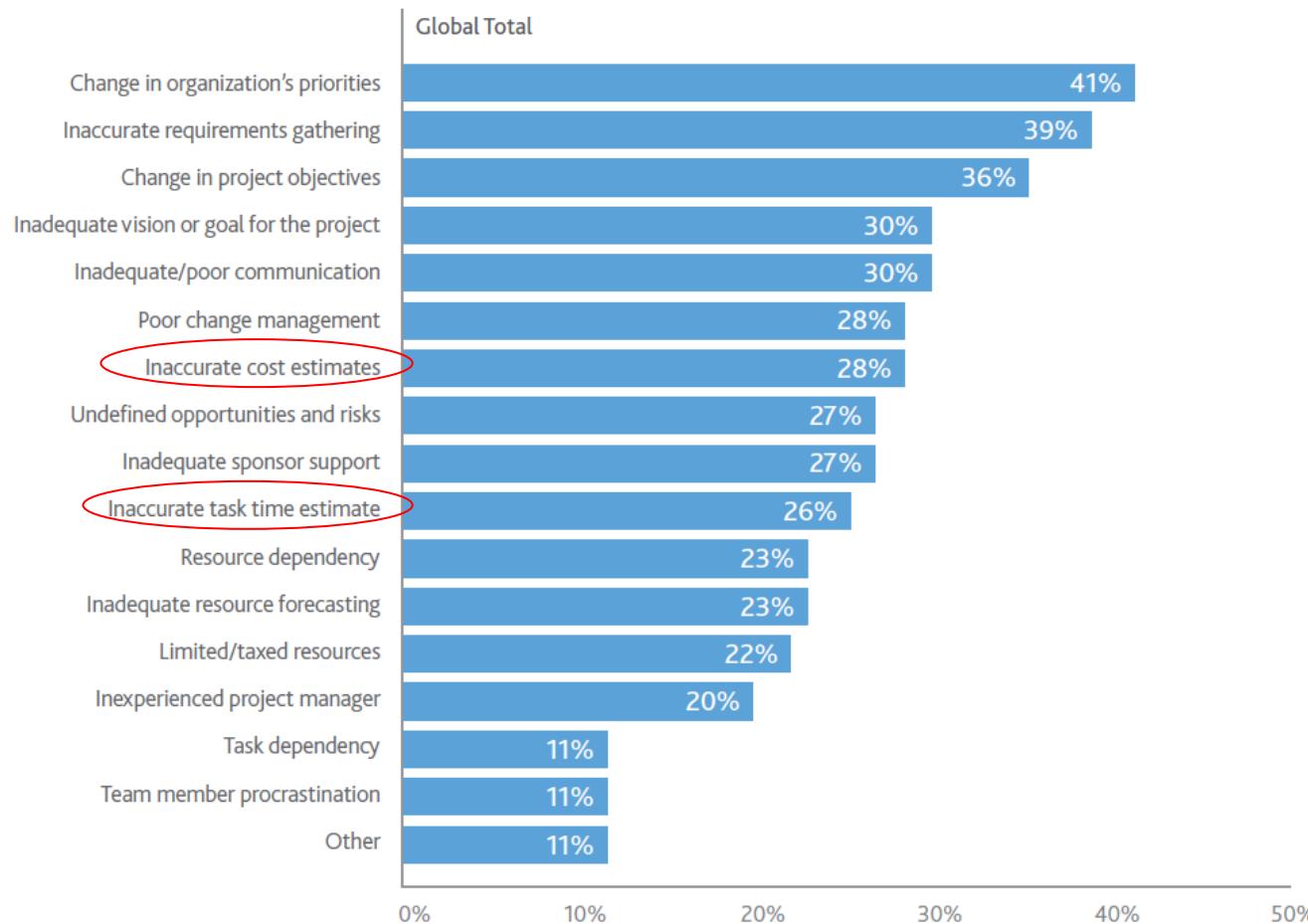
Why do we need to improve effort estimation?



Source: Literature Review PhD Student Jirat Pasuksmit



Q: Of the projects started in your organization in the past 12 months that were deemed failures, what were the primary causes of those failures? (Select up to three.)



PMI's PULSE of the PROFESSION -<https://www.pmi.org/learning/thought-leadership/pulse/pulse-of-the-profession-2017>



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand a range of software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation



1. Expert judgement

- Several experts on the proposed software development technique and the application domain estimate project cost. These are then discussed, compared and adjusted until consensus is reached
- Some expert judgement techniques involve polling each expert independently, in some cases for three estimates, pessimistic estimate (p), optimistic estimate (o) and the most likely estimate (m), and the expert's estimate is computed as the:

$$e = (p + 4m + o)/6$$

- *Delphi technique*: asks several experts to make an individual judgement of the effort using any method they wish. Then, the average effort is calculated, and presented to all of the experts. Each expert is then given a chance to revise their estimate, in some cases after a discussion between all experts. This continues until no expert wishes to revise their estimate.



SWEN90016

2. Estimation by Analogy

- The cost of a new project is estimated based on similar projects in the same application domain

3. Parkinson's Law

- This law states that the work will expand to fill the time available
- The cost is determined by available resources rather than by objective assessment
- For example, if the software is to be delivered in 12 months, and 3 people are available, the effort is 36 person months

4. Pricing to win

- The cost is estimated to be whatever the customer has available to spend on the project - cost depends on the budget not on the software functionality



5. Algorithmic cost modelling

- A **model** is developed using **historical cost information** based on some **software metric (usually its size)** to the project cost
- When a project effort needs to be estimated, an estimate of the metric is computed
- Using the model, the effort is predicted
- The most general form of an algorithm cost estimate is given by:

$$Effort = A \times Size^B \times M$$

- A** - a constant factor that depends on the organizational practices
- Size** - size of the software estimated in a metric of choice (e.g. lines of code, function point, use case points)
- B** - a value between 1 and 1.5 derived experimentally
- M** - a multiplier made by combining process, product and development attributes such as stability of requirement, experience of the team



WILLING TO LEARN

5. Algorithmic cost modelling - cont...

$$Effort = A \times \text{Size}^B \times M$$

Basic steps in algorithmic cost estimation

1. Estimate the *size* of the development product
2. Estimate the *effort in person-months* or person-hours
3. Estimate the *schedule in calendar months*
4. Estimate the *project cost in agreed currency*



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation



- Commonly used metric for software size estimation
 - Source Lines of Code (SLOC)
 - Based on code
 - Function Points (FP)
 - Based on the Requirements Specification
 - Use-case Points (UCP)
 - Based on Use Cases



WILLING TO LEARN

- There are two types of SLOC:
 - **Physical SLOC:** Count the number of lines excluding comments and blank lines
 - **Logical SLOC:** Measure the number of executable "statements", but their specific definitions are tied to specific computer languages

C	COBOL
<pre># include <stdio.h> int main() { printf("\nHello world\n"); }</pre>	<pre>identification division. program-id. hello . procedure division. display "hello world" goback . end program hello .</pre>
Lines of code: 4 (excluding whitespace)	Lines of code: 6 (excluding whitespace)



WILLING TO LEARN

- Advantages of SLOC:
 - Scope for Automation of Counting: Since Lines of Code is a physical entity it is easy to count and can be automated using a tool
 - An Intuitive Metric: Lines of Code serves as an intuitive metric for measuring the size of software because it can be seen and the effect of it can be visualized



SWEN90016

- Disadvantages of SLOC:

- Variability: Depends on programmer experience, programming language, framework support (auto generated code), reuse, etc.
- It is difficult to estimate the number of lines of code that will be needed to develop a system from information that is available in analysis and design phases
- Lack of a universally accepted definition for exactly what a line of code is



- Is used to express the *amount of functionality* in a software system, as seen by the user
- A *higher number of function points* indicates *more functionality*
 - Empirical evidence demonstrates that there is a *positive correlation between function points and the complexity of the system*
- Typically used to:
 - Estimate the cost and effort required to design, code and test a software system
 - Predict the number of errors
 - Predict the number of components
 - Measure productivity
- Function points are computed from the *Software Requirements Specification (SRS)*



- **Advantages of Function Points**

- Measures the size of the solution instead of the size of the problem
- Requirements are the only thing needed for function points count
- Can be estimated early in analysis and design
- Is independent of technology
- Is independent of programming languages



- **Disadvantages of Function Points**

- A well defined requirements specification is necessary
- Gaining proficiency is not easy, the learning curve is quite long
- Could be quite time-consuming thus could be costly

We will be going through an example



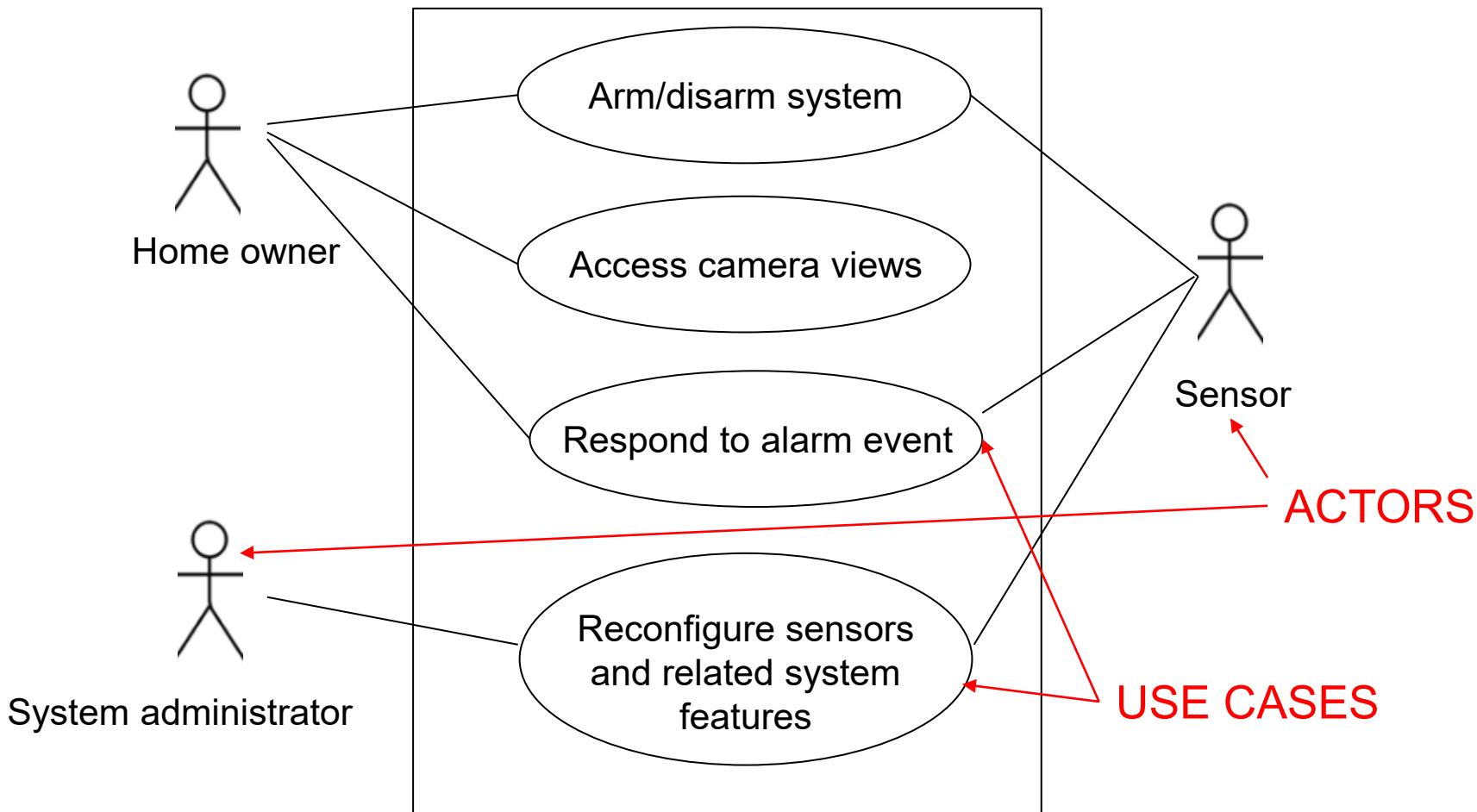
- Is a software estimation technique used to measure the software size with Use Cases
- Developed in 1993 for sizing and estimating projects using OO methodology
 - developed by Gustav Karner of Objectory (now Rational Software)
- The concept of UCP is similar to FPs



- Use cases describe the functionality of the system.
- Use cases model the dialog between the actors and the system.
- Primary purpose is to document functional requirements but also used for testing purposes,



WILLING TO LEARN



Use Case Diagram for a SafeHome Security System



WILBERFORD

Use Case: Access camera views

Description: To view output of cameras from any remote location via the Internet

Primary Actor: Home owner

Preconditions: System must be fully configured; appropriate use-id and password must be available

Trigger: Home owner decides to view the camera output while away.

Scenario:

1. The home owner logs onto the SafeHome system via the Internet.
2. The home owner enters the user ID and password.
3. The system display major function buttons.
4. The home owner selects 'surveillance' from the major function buttons.
5. The home owner selects 'pick a camera' option.
6. The system displays the available cameras on a map.
7. The home owner selects the camera of interest.
8. The home owner selects the 'view' button.
9. The system display the video output on the viewing window.

Extensions:

- 2a. ID or password are incorrect; user is requested to re-enter the password or to validate password.
 - 4a. Surveillance function is not properly configured; system displays an error message and exits.
 - 6a. Map is not properly configured; system request the user to configure the map.
-



1. Compute Unadjusted Use Case Weight (UUCW)
2. Compute Unadjusted Actor Weight (UAW)
3. Compute Technical Complexity Factor (TCF)
4. Compute Environmental Complexity Factor (ECF)
5. Compute the final size estimate



WILLING TO LEARN

1. Compute Unadjusted Use Case Weight (UUCW)

Count the number of simple average, complex use cases, N_s, N_A, N_C based on the number of transactions as per table below.

- The number of transactions can be computed by counting the number of steps in the scenario

Use Case Classification	Type of Actor	Weight
Simple	1 to 3 transactions	5
Average	4 to 7 transactions	10
Complex	8 or more transactions	15

$$UUCW = N_s \times 5 + N_A \times 10 + N_c \times 15$$



WILLING TO LEARN

2. Compute Unadjusted Actor Weight (UAW)

Count the number of simple average, complex actors, N_s, N_A, N_C as per table below.

Actor Classification	Type of Actor	Weight
Simple	External system interacting using a well defined API	1
Average	External system interacting using a standard protocol (e.g. TCP/IP, FTP, HTTP)	2
Complex	Human actor using a GUI	3

$$UAW = N_s \times 1 + N_A \times 2 + N_c \times 3$$



3. Compute Technical Complexity Factor (TCF)

Factor	Description	Weight
T1	Distributed system	2.0
T2	Response time	1.0
T3	End-user efficiency	1.0
T4	Internal processing complexity	1.0
T5	Code reusability	1.0
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portability to other platforms	2.0
T9	System maintenance	1.0
T10	Concurrent/parallel processing	1.0
T11	Security features	1.0
T12	Access for third parties	1.0
T13	End user training	1.0

Score each factor between 0 – 5

- 0 - **irrelevant**
- 5 - **essential**

$$TF = \sum_{i=1}^{13} S_i \times W_i$$

S_i - Score for the i -th factor
 W_i - Weight of the i -th factor

$$TCF = 0.6 + TF/100$$



WILLING TO LEARN

3. Compute Technical Complexity Factor (TCF)

Factor	Description	Weight	Score
T1	Distributed system	2.0	0
T2	Response time	1.0	5
T3	End-user efficiency	1.0	4
T4	Internal processing complexity	1.0	3.0
T5	Code reusability	1.0	0
T6	Easy to install	0.5	1
T7	Easy to use	0.5	2
T8	Portability to other platforms	2.0	1
T9	System maintenance	1.0	3
T10	Concurrent/parallel processing	1.0	4
T11	Security features	1.0	5
T12	Access for third parties	1.0	1
T13	End user training	1.0	3

Score each factor between 0 – 5

- 0 - irrelevant
- 5 - essential

$$TF = \sum_{i=1}^{13} S_i \times W_i$$

S_i - Score for the i -th factor
 W_i - Weight of the i -th factor

$$TCF = 0.6 + TF/100$$



WILLING TO LEARN

4. Compute Environmental Complexity Factor (ECF)

Factor	Description	Weight
E1	Familiarity with development process used	1.5
E2	Application experience	0.5
E3	Object-oriented experience of team	1.0
E4	Lead analyst capability	0.5
E5	Motivation of the team	1.0
E6	Stability of requirements	2.0
E7	Part-time staff	-1.0
E8	Difficult programming language	-1.0

Score each factor between 0 – 5

- 0 - **irrelevant**
- 5 - **essential**

$$EF = \sum_{i=1}^8 S_i \times W_i$$

S_i - Score for the i -th factor
 W_i - Weight of the i -th factor

$$ECF = 1.4 + (-.03 * EF)$$



WILLING TO LEARN

5. Compute the final size estimate

$$UCP = (UUCW + UAW) \times TCF \times ECF$$



- **Advantages of Use Case Points**

- UCPs are based on use cases and can be measured very early in the project life cycle
- UCP based estimates are found to be close to actuals when estimation is performed by experienced people
- UCPs are easy to use and do not call for additional analysis
- Use cases are being used vastly as a method of choice to describe requirements



- **Disadvantages of Use Case Points**

- UCP can be used only when requirements are written in the form of use cases
- Dependant on goal-orientated, well written use cases
- Technical and environmental factors have a high impact on UCP
- Not as well established as FPs



Lecture Break

BREAK

Please return promptly as the

Lecture will re-start in *5 mins*



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation

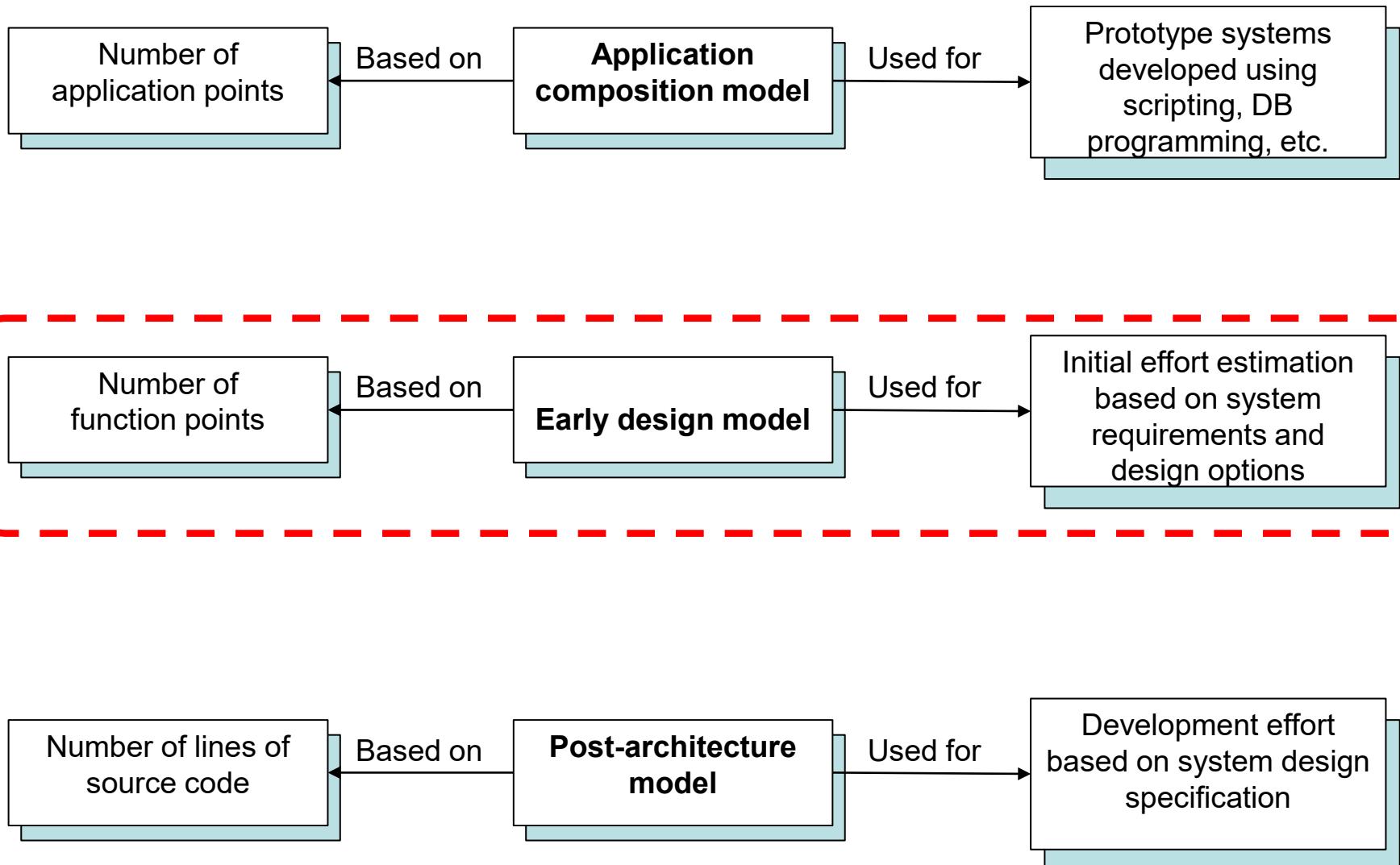


SWEN90016

- Derived from collecting data from a large number of software projects and deriving formulae that best fits the observations
 - *an empirical model*
- It has been widely used and evaluated in a range of organizations
- Well documented, available in the public domain and is well supported by tools
- It has been in use for a long time:
 - first proposed in 1982 (Boehm_1981)
 - most recent version, COCOMO II was published in 2000 (Boehm_2000)



WILLING & ABLE





$$\text{Effort} = A \times \text{Size}^B \times M$$

A - a constant factor that depends on the organizational practices

Size - size of the software estimated in a metric of choice

B - a value between 1 and 1.5 derived experimentally

M - a multiplier made by combining process, product and development attributes such as stability of requirement, experience of team

Effort - total effort measured in person-months

In the COCOMO II early design model:

Parameter	Value	Notes
A	2.94	Estimated empirically
Size	KSLOC (thousands of lines of code)	Computed from FPs
B	1.01 – 1.26	Explained later
M	Computed based on project and process characteristic	Explained later



WILLING TO LEARN

COCOMO II: Estimating *Size* in KSLOC ($Effort = A \times Size^B \times M$)

- Size is estimated based on logical lines of code
- Can be estimated based on FPs using the table below

Language	Average	Median	Low	High
Ada	154	-	104	205
Assembler	209	203	91	320
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	66
Fortran	90	118	35	-
Java	55	53	9	214
Perl	57	57	45	60
Visual Basic	50	52	14	276

Number of logical lines of code per FP

e.g. 200 FPs: in C = $29.6 \text{ (} 200 \times 148 / 1000 \text{)} \text{ KSLOC}$; in Java = $11 \text{ (} 200 \times 55 / 1000 \text{)} \text{ KSLOC}$
(per 1000 lines of code)



WILLING TO LEARN

COCOMO II: Estimating parameter B ($Effort = A \times Size^B \times M$)

$$B = 1.01 + .01 \sum_{i=1}^5 W_i$$

W_i - a scaling factor value ranging from 0-5 as per table below.

Precedentedness	Familiarity of the application domain: 0 (thoroughly familiar) to 5 (completely unprecedented).
Development flexibility	The level of flexibility in development process, methods, and tools, ranging from 0 (general goals) to 5 (rigorous).
Architecture completed and risks eliminated	20%: 5; 40%: 4; 60%: 3; 80%: 2; 90%: 1; 100%: 0
Team cohesion	The level of interaction within the team. Ranging from 0 (seamless interactions) to 5 (very difficult interactions).
Process maturity	A ranking use the Software Engineering Institute's Capability Maturity Model, ranging from 0 (chaotic, following no processes) to 5 (actively measuring and optimising processes) (Quality Management)



WILLING TO LEARN

COCOMO II: Estimating M ($Effort = A \times Size^B \times M$)

- the cost drivers consist of seven different factors
- each factor is rated on a six point scale; *very low* to *extra high*

$$M = RCPX \times RUSE \times PDIF \times PREX \times PERS \times SCED \times FCIL$$



© THE UNIVERSITY OF MELBOURNE

RCPX	The expected complexity of the internal processes, and the level of reliability required for the system.
RUSE	The level of reuse that this code developed in this system is expected to offer to future systems.
PDIF	The level of platform difficulty. This refers to the constraints placed on the system by the platform on which it runs, such as the amount of processor time and storage available.
PREX	The experience of the personnel on the project. Ranging from less than 2 months (very low) to more than 6 years (very high).
PERS	The capability of the personnel on the project. Ranging from the 15th percentile (very low) to the 90 percentile (very high).
SCED	The constraints placed upon the project schedule, rated as a percentage of the “stretch-out” of the schedule. Schedules that are highly compressed (not stretched-out), require more effort than the optimal to complete the project on time. A rating of very low is a schedule that is 75% the length of the nominal project. A rating of very high is a project schedule that is 160%+ of the nominal. Empirical evidence suggests that compressing the schedule, and therefore expending less effort, results in lower quality software.
FCIL	The team support facilities.



WILLING TO FIGHT

Cost Driver	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
RCPX	0.75	0.88	1.00	1.15	1.30	1.66
RUSE		0.91	1.00	1.14	1.29	1.49
PDIF		0.87	1.00	1.11	1.27	1.62
PREX	1.23	1.11	1.00	0.89	0.82	
PERS	1.37	1.16	1.00	0.87	0.75	
SCED	1.29	1.10	1.00	1.00	1.00	1.00
FCIL	1.24	1.11	1.00	0.89	0.79	0.78

Cost driver ratings for the COCOMO II early-design phase model.



COCOMO II: Estimating Effort

- Based on the computed parameter values the effort can be estimated:

$$Effort = A \times Size^B \times M$$

COCOMO II: Estimating Time (T) and Number of Personnel (N)

- Formula for estimating the nominal delivery time:

$$T = 2.5 \times Effort^{(0.33+0.2(B-1.01))}$$

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



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation
5. Understand cost estimation techniques used in Agile software development lifecycles



Two key concepts that are used to effort estimation:

Story points: a story point is a relative measure of the size of a user story (recall that the requirements of the system are documented using user stories)

Velocity: velocity is a measure of productivity of team, which is represented by the number of story points delivered in a specified time period



1. Develop user stories for the system.
2. Estimate the number of story points for each story, basing the estimate on the number of story points from previous stories, using a chosen technique (discussed later).
3. Use the team's velocity from previous experience to estimate the delivery time of the project - in the case of fixed-scope release planning develop a release burn-down chart.
4. During development, measure the actual velocity of the team.
5. Using this velocity, re-estimate the time it will take to deliver the product.



- **Estimate by analogy**

- There are no units for story points, always base our measures on other stories. If story A is about the same size as story B, they should have the same number of story points.

- **Decompose a story**

- By decomposing a story into the tasks that are required to complete the story, we can find measures that we know about the tasks, and combine them to provide a total measure.

- **Use the right units**

- The relative units should not be too fine grained. A pattern-based scale is used. For example, measures can only be 1, 2, 4, 8, or 12 or numbers in the Fibonacci sequence.

- **Use group-based estimations**

- For a story that is to be implemented by a team, the whole team should provide estimates. Techniques such as the Delphi method or its adaptations can be used to reach consensus.



SWEN90016

- Agile Estimation Techniques:

- Planning Poker
- Bucket System
- Relative Mass Valuation
- T-Shirt Sizes
- Affinity Estimation
- Dot Voting



WILLING TO RISK

1. Customer reads story.



Development team asks questions

2. Team estimates.
This includes testing.



Discussion ...

3. Team discusses.



4. Team estimates again.
Repeat until consensus reached.



<https://www.sitepoint.com/3-powerful-estimation-techniques-for-agile-teams/>



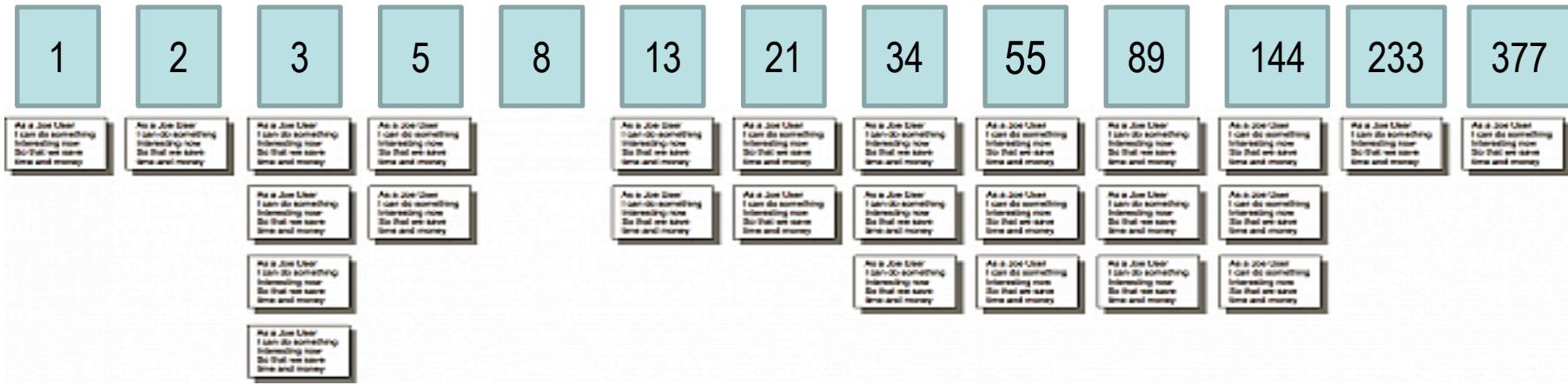
WILLING TO FIGHT



<http://www.agileadvice.com/2013/07/30/referenceinformation/agile-estimation-with-the-bucket-system/>



MILLION OF FIVE



- The team sitting at a table picks a user story card randomly and places it in bucket 8
- The next few cards are randomly picked one at a time, discussed agreed on, and placed in a bucket relative to the previous ones
- Then each person is allocated a set of cards and they are placed in a appropriate bucket, based on individual judgement (Divide and conquer)
- Finally the team reviews the placements and reach agreement



WILLING TO RISK

1. Set up a large table so the stories can be moved around easily relative to each other.
2. Pick any story to start, team estimates whether they think that it is relatively: Large, Medium, Small.
3. Large story one end on the table. Medium story in the middle and Small story the other end
4. Continue through steps 2 & 3
5. The next step is to assign points values based on the position of the stories on the table. Start with the easiest story that is worth assigning points to, and call it a 1.
6. Then move up the list of cards, assigning a value of 1 to every story until you get to one that seems at least twice as difficult as the first one. That story gets a 2.

<http://www.flowless.eu/relative-mass-valuation/>



WILLING TO LEARN

$$V = \frac{SP}{T_i}$$

V - velocity

SP - number of story points completed

T_i - time period over which they were completed

Common methods for measuring velocity:

- Using historical data
- Using data from previous iterations



$$T = \frac{\sum_{i=1}^n SP_i}{V}$$

T - estimate delivery time

V - velocity

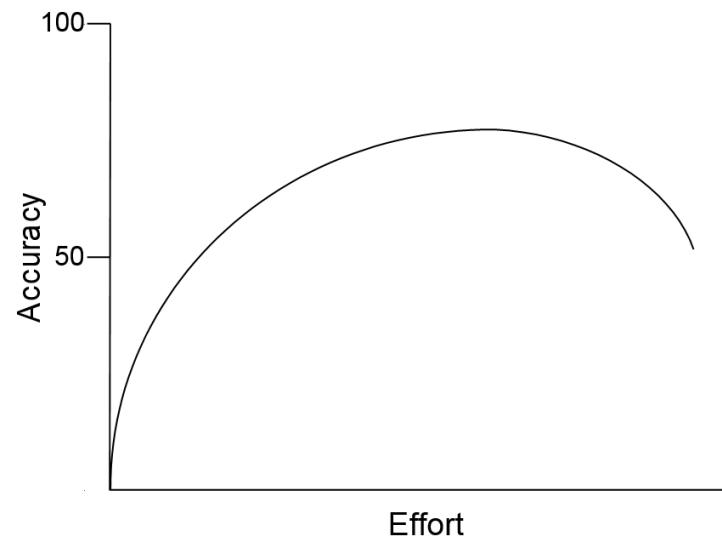
SP_i - number of story points in the i -th user story

n - total number of user stories



WILLING TO RISK

- Allow enough time to do a proper project estimate - rushed estimates are inaccurate, high-risk estimates
- There is diminishing return on time spent estimating



- Agile teams choose to be closer to the left
- Know that you cannot eliminate uncertainty from estimates but small efforts are rewarded with big gains



1. Understand the importance of cost estimation and the challenges involved
2. Understand the techniques used for cost estimation
3. Understand software size estimation techniques
4. Understand the principles of the COCOMO II model for algorithmic cost estimation
5. Understand cost estimation techniques used in Agile software development lifecycles

References

1. B. Boehm, C. Abts, W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. Software Cost Estimation with Cocomo II. Prentice Hall, 2000.
2. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
3. I Somerville. Software Engineering, Addison-Wesley Publishing, ninth edition, 2010.
4. K. Molkken and M. Jrgensen, A Review of Surveys on Software Effort Estimation, Proceedings of the 2003 International Symposium on Empirical Software Engineering

cont...



References

5. Danh Nguyen-Cong and De Tran-Cao, A review of effort estimation studies in agile, iterative and incremental software development, The 2013 RIVF International Conference on Computing Communication Technologies - Research, Innovation, and Vision for Future (RIVF)
6. M. Usman, E. Mendes and F. Weidt and R. Britto, Effort Estimation in Agile Software Development: A Systematic Literature Review, Proceedings of the 10th International Conference on Predictive Models in Software Engineering, 2014
7. PMI's PULSE of the PROFESSION -<https://www.pmi.org/learning/thought-leadership/pulse/pulse-of-the-profession-2017>

1. B. Boehm, C. Abts, W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. Software Cost Estimation with Cocomo II. Prentice Hall, 2000.
2. R. S. Pressman. Software Engineering: A Practitioner's Approach. McGraw Hill, seventh edition, 2009.
3. I Somerville. Software Engineering, Addison-Wesley Publishing, ninth edition, 2010.
4. A Review of Surveys on Software Effort Estimation
5. A review of effort estimation studies in agile, iterative and incremental software development, The 2013 RIVF International Conference on Computing Communication Technologies - Research, Innovation, and Vision for Future (RIVF)

SOFTWARE ENGINEERING PROCESSES AND MANAGEMENT



MEET THE CASE STUDIES

INTRODUCTION TO THE CASE STUDIES

The aim of the case-studies is to provide concrete, real-world examples of how software engineering principles can be applied. That is to say that the aim of studying cases is to help you develop the *practice of software engineering* by studying what others have done.

Throughout this subject we will be using two case studies. The first is a Virtual Reality (VR) simulator for training surgeons to perform temporal bone and general mastoid surgery. This will largely be used for assessment items throughout the semester. The second is a web-based framework that allows users to share language data, language resources and application modules across the web. This will be used in lectures and workshops to demonstrate concepts.

These two cases studies represent two different kinds of systems, each requiring different approaches, tools and techniques. The point of software engineering is to *design* processes that build in the most appropriate approaches, tools and techniques so that each type of project will meet its goals, and in turn be a success.

The virtual reality simulation requires a software system that will integrate a number of devices as well as simulate a real world task. It combines computer graphics and embedded systems and must:

- be reliable¹;
- meet strict performance criteria²; and
- be usable.

The types of processes, tools and techniques needed here are those pertinent to measuring and evaluating software reliability and performance³ and those pertinent to assessing the usability of the Human Machine Interface (HMI).

On the other hand the web framework has different characteristics for example, it must:

- be extensible;
- be usable; and
- maintain the integrity of its data across different formats.

The types of processes, tools and techniques needed here are those pertinent to evaluating distributed systems, networks and network latencies. Note that there is also a need to choose processes, tools and techniques pertinent to the usability of the interface, but it is a different interface to the one for the VR simulation. The reason is that the type of tasks being undertaken a more centred around the interchange of knowledge and applications while the VR simulations is centred around enacting a real-world task.

Even when two characteristics are apparently similar we still find that we require different approaches or different tools and techniques.

¹We will encounter the term *reliability* quite frequently, however, in software engineering it means something specific — it is the probability of the system *not* failing for a given time.

²In this project, the term *performance* is taken to mean the actual execution time of the system running on a machine. We must distinguish carefully between terms like *performance*, *efficiency* (which is a more general term), and *algorithmic complexity*.

³We will meet these types of processes, tools and techniques in the subject *Software Engineering Methods*.

CASE STUDY 1 — VIRTUAL TEMPORAL BONE SURGERY

INTRODUCTION

The problem in the first case study is to develop a simulator for temporal bone surgery. Temporal bones are found at the side of the head, as shown in Figure 1.

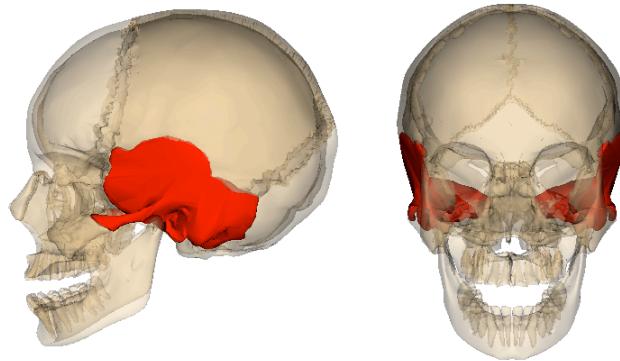


Figure 1: An image of the skull with the temporal bone shaded.

There is a drive in surgical training towards giving trainee surgeons more practice at standard procedures and also to expose trainees to a more diverse caseload that gives them some experience of non-standard procedures. The traditional apprenticeship model of training surgeons has been criticised for not delivering the kinds of in-depth focussed training nor the diversity in the operative caseload that helps trainee surgeons become more competent at their craft.

There are a number of methods for giving trainee surgeons the extra practice that they need, but few for giving them the *diversity* that they need. One of the most promising methods for providing that with diverse experience is *virtual reality* (VR) computer-based simulations.

The problem in the first case study is the development of exactly such a virtual reality simulator. The simulator will be for:

1. the development of surgical skills in drilling the temporal bone; and
2. learning surgical decision making by being exposed to different cases of temporal bone surgery.

The VR simulator will require several 3-D visual models of the temporal bone, especially the internal anatomy of the temporal bone including the facial nerve, the sigmoid sinus, the porous bone of the temporal bone, the major aircells in the temporal bone, the incus, and the middle ear. It will also require a force model for the amount of force required to drill bone.

All this makes the VR simulation project interesting for a number of reasons.

1. The software must be able to reproduce a real-world task — the simulation — but how close can we get? The point here is that we will need to do some non-standard systems analysis to get to a system that will be *fit for purpose*.

2. The software will need to make several new and interesting pieces of hardware work together *reliably* and according to strict *performance constraints* that will make this software system usable.
3. The final system will be used by trainee surgeons, not computing experts. It must, therefore, be intuitive to use and easy to learn.
4. It will require some interesting algorithms to compute the forces, changes to model and responses of the system.

THE PROBLEM STATEMENT

The problem of training otalaryngology surgeons to be safe and competent in the insertion of cochlea implants is becoming critical because of the shortage of quality cadavers on which to practice.

The problem affects the training of surgeons, the quality of care provided to patients, and the medical indemnifiers, because of the risk of poorly trained surgeons undertake procedures. It also affects the manufacturers of implants because of the perceived risk among the population.

A computer aided virtual surgery environment would benefit not only the public by having surgeons who are more practised with the procedure, but also the companies selling implants, indemnifiers and surgeons by lowering risk.

KEY FEATURES OF THE SOLUTION: A VR TEMPORAL BONE SURGERY SIMULATOR

The aim of the project is to build a virtual reality haptic training simulation for temporal bone and general mastoid surgery. Note the word *haptic* relates to the sense of touch and the perception of manipulating objects through the sense of touch. The sense of touch is delivered through haptic input/output devices such as those in Figure 2. When a trainee touches a piece of bone, the haptic device applies resistance against the trainee's hand to simulate touching real bone.



Figure 2: Force input/output devices by Sensable technologies.

Assume for now that the key features of the solution that will make it *fit for purpose* are the following.

1. A *load* feature that allows the surgeon to load one or several temporal bone models to work on.
2. A *3D stereoscopic rendering* feature where the model can be viewed in 3D.
3. A *3D viewing feature* that allows the user to rotate the model about various axes, view the model from different viewpoints, to zoom in and out, and to view the model as a surgeon would through a microscope
4. A *select surgical procedure* feature that does the following:
 - (a) allows the user to select a specific type of surgical procedure, such as cochlea implant, or mastoidectomy;
 - (b) then to provide the user with an interface that shows at least the surgical tools, a model, and a set of parameters for the procedure; and
 - (c) allows the user to configure the haptic IO device for the procedure.
5. A *perform surgical procedure* feature that allows a user of the system to simulate a surgical procedure by:
 - (a) interacting with the model via the haptic device and other IO devices; and
 - (b) having the model updated to reflect the impact of the user's actions.
6. A *save and restore* feature that allows the surgeon to save the current state of the simulation and restore it later.
7. A *replay and pause* feature that allows the surgeon to replay their interaction with the simulation and to see and feel all of their actions as well as pausing and rewinding the simulation.

AN INITIAL LIST OF STAKEHOLDERS

The initial group of stakeholders represent the collection of people and organisations relevant to the project at the project concept phase — the phase of the project at the very start before any real development activity has taken place. The stakeholder analysis may be expanded to take in other stakeholders as we uncover more about the project.

1. *The clients* are two otolaryngology surgeons from the Royal Victorian Eye and Ear hospital (RVEEH) engaged in both surgical practice and training of surgeons in temporal bone and mastoid surgery.
It is expected that the surgeons will be able to provide more comprehensive training for their trainee surgeons overcoming some of the problems in today's training methods. It is also expected that the two surgeons will enhance their own standing by undertaking cutting edge projects.
2. *The development team* is a group of 12 relatively inexperienced software engineers. The team has experience in Java programming, but not an extensive experience in the Java environment, for example, they have only a limited knowledge of the Java libraries for image processing and graphics (perhaps you may want to think of them as 4th year software engineering students).
3. *A management team* consisting of two experienced people from the developer's organisation (perhaps a good way to think of these two people is as the supervisors and coordinators from the University of Melbourne).

4. *Trainee otolaryngology surgeons* who are learning the procedures pertaining to the temporal bone. Interestingly these trainee surgeons are not complete novices because in order to get into otolaryngology they must have at least a general background in surgery.
5. *The IT team at the RVEEH* who will need to maintain the system and take it through their hardware and software upgrades. The main concern of most IT people is that they can manage the major changes to their systems as effortlessly as possible. They certainly don't want the software to break and cause problems that they can't fix.

SOFTWARE ENVIRONMENT AND TECHNOLOGY

The software and technology environment may cause some difficult problems in this project. The original idea for the project brings together:

1. programming in C++;
2. programming 3D models of the human temporal bone;
3. rendering models for 3D stereoscopic viewing using 3D glasses;
4. rendering force models through the haptic device; and
5. maintaining the state of a surgical procedure.

For the development team involved there are quite a number of challenges: (1) the development team is largely unfamiliar with C++; (2) the development team is unfamiliar with modelling and rendering 3D stereoscopic models; and (3) the team has had little experience in haptics and programming for haptic feedback devices.

In the concept phase the original idea was to use C++ because that was more acceptable to the IT department at the RVEEH. It is not an absolute requirement though and can still be negotiated.

For this project, technology may become a major factor and so your project team needs to think about technology very carefully...!

CASE STUDY 2 — A WEB FRAMEWORK FOR LANGUAGE RESEARCH

INTRODUCTION

Research in human languages, ranging from the preservation of endangered dialects through to the use of strategies in collaborative language learning, is often stalled because it is time consuming, idiosyncratic and prone to error. The use of computers for data recording, archiving and as a medium for interaction promises to revolutionise how investigations are conducted and, indeed, offer venues for research into the application of computers in second language acquisition.

The problem with the few computer-based systems in existence are that they are not applicable beyond the limited contexts in which they were built and often must be used within institutions and within institutional aims. The issue is complicated by the fact that the majority of efforts tend to lack sound requirements, mature software engineering practice, clear documentation, and wide interoperability.

This project aims to deliver a web-based framework that can overcome the problems noted above, meet the complex demands of language technology research and education across the globe and be extensible.

This project is interesting because there are the following problems that are related to it being a distributed system:

1. There is the problem of data integration from a number of distributed resources in a number of different formats. One of the features of the system must be to take care of the differences between local — and idiosyncratic — data formats.
2. There is the problem of sharing processing modules in a distributed environment. Should modules be stored local and access remotely — for example by a remote method invocation — or should multiple copies of applications be spread around the network?
3. There is the problem of control. Since this is a distributed system and is intended to be peer-to-peer there is no obvious server that can act as a single point of control.

PROBLEM STATEMENT

The problem of sharing language resources is to allow researchers to share their language resources and processing modules on a global scale.

The problem affects the progress of research, the limiting of language research results to within local communities, the duplication of research and consequently funding and the progress of the discipline.

A web-based system would benefit research community by providing a key point of contact for resource sharing including data and analysis modules as well as a key focal point for the dissemination of the latest work.

KEY FEATURES OF THE SOLUTION: WEB-BASED LANGUAGE RESEARCH FRAMEWORK

The framework will need to provide the following features.

1. A *data integrity* feature that assures the integrity of the applications and data shared within the system and its community.
2. A *user groups* feature that enables certain data and applications to only be shared among a certain subset of users. The system must be able to create user groups, handle membership and assure that data and applications that are registered to a specific user group are used only by users within that group.
3. A *registration of user data and applications* feature that allows registered system users to add their own data and applications to the system for use by the community.
4. A *searching* feature that allows users to setup criteria for searching and to retrieve data and applications via searching.
5. A *transfer* feature that facilitates the transfer of data or applications registered at one location to another location.
6. A *data format translation* feature that supports the conversion of data between certain data formats and internal system formats. The framework should know, from a profile, what data formats are required, and should perform data conversion automatically.

AN INITIAL LIST OF STAKEHOLDERS

As before we have an initial list of stakeholders for the system. We can again expand this list or refine the outcomes for each stakeholder once we know more about the project.

1. *The client* who is a language and technology researcher at the University of Melbourne.
2. *The development team* is a group of 12 relatively inexperienced software engineers. The team has experience in Java programming, but not an extensive experience in the Java environment, for example, they have only a limited knowledge of the Java libraries for image processing and graphics (perhaps you may want to think of them as 4th-year software engineering students).
3. *A management team* consisting of two experienced people from the developer's organisation (perhaps a good way to think of these two people is as the supervisors and coordinators from the University of Melbourne).
4. *The users* of the system will be academics, researchers or language learners who will use the system to share and acquire applications and data resources. They are assumed to be capable of using a PC and to be able to competently use a web browser, register and share resources, and perform search and download activities relevant resources.
5. *The administrators* of the system are unknown at this point. How the system will be administered needs to be decided as requirements. Due to the distributed nature of the system, there will need to be a way for an administrator on one node to fix problems at another node.

inspiring success



**ACS Code of Professional Conduct
Professional Standards Board
Australian Computer Society**

April 2014

ACS Code of Professional Conduct

Version History

Date	Document Version	Revision History (reason for change)	Author /Reviser
06.07.2012	V.2	Corporate Identity Changes	Ruth Graham
04.04.2014	V2.1	Removed reference to Code of Professional Practice	Graham Low

Approvals

Date approved	Version	Approved By	Date in force	Date of Next Review
2005	V.1	Professional Standards Board	2005	To be confirmed
2014	V2.1	Michael Johnson	4 April 2014	To be confirmed

Custodian title & e-mail address:	andrew.johnson@acs.org.au
Responsible Business Group:	Professional Standards Board
Distribution:	General (no restriction on distribution)
Content Security:	Unclassified

TABLE OF CONTENTS

ACS Code of Professional Conduct.....	
1.1. PREAMBLE.....	4
Relevance To Law	4
1.2. THE CODE.....	5
1.2.1. The Primacy of the Public Interest	6
1.2.2. The Enhancement of Quality of Life.....	6
1.2.3. Honesty	7
1.2.4. Competence	7
1.2.5. Professional Development	8
1.2.6. Professionalism	8

1.1. PREAMBLE

An essential characteristic of a profession is the need for its members to abide by a code of ethics. For the ACS, this code is established as the Code of Professional Conduct and is part of the Society's Regulations. These Regulations apply to any ACS member who works in the field of information and communications technology (ICT).

This Code of Professional Conduct (the Code) identifies six core ethical values and the associated requirements for professional conduct. The Society requires its members to abide by these values, and act with responsibility and integrity in all of their professional dealings.

Relevance to Law

This Code of Professional Conduct has relevance to professional standards legislation. Failure to abide by the Code could be used as grounds for a claim of professional negligence. The Code may be quoted by an expert witness giving an assessment of professional conduct. Failure to observe the Code could also lead to disciplinary action by the ACS.

1.2. THE CODE

As an ACS member you must uphold and advance the honour, dignity and effectiveness of being a professional. This entails, in addition to being a good citizen and acting within the law, your conformance to the following ACS values.

1. ***The Primacy of the Public Interest***

You will place the interests of the public above those of personal, business or sectional interests.

2. ***The Enhancement of Quality of Life***

You will strive to enhance the quality of life of those affected by your work.

3. ***Honesty***

You will be honest in your representation of skills, knowledge, services and products.

4. ***Competence***

You will work competently and diligently for your stakeholders.

5. ***Professional Development***

You will enhance your own professional development, and that of your staff.

6. ***Professionalism***

You will enhance the integrity of the ACS and the respect of its members for each other.

In a situation of conflict between the values, The Primacy of the Public Interest takes precedence over the other values.

This Code of Professional Conduct is aimed specifically at you as an individual practitioner, and is intended as a guideline for your acceptable professional conduct. It is applicable to all ACS members regardless of their role or specific area of expertise in the ICT industry.

The following list of requirements is not exhaustive and should not be read as a complete definition of acceptable professional conduct in all practical situations. The intention of the Code is to illustrate what constitutes professional behaviour. You are expected to take into account the spirit of this Code in order to resolve ambiguous or contentious issues concerning professional conduct. The ACS can help you resolve ethical dilemmas.

1.2.1. The Primacy of the Public Interest

In the context of this Code, the public interest takes precedence over personal, private and sectional interests, and any conflicts should be resolved in favour of the public interest. In your work, you should safeguard the interests of your immediate stakeholders, provided that these interests do not conflict with the duty and loyalty you owe to the public. The public interest is taken to include matters of public health, safety and the environment.

In accordance with this value you will:

- a) identify those potentially impacted by your work and explicitly consider their interests;
- b) raise with stakeholders any potential conflicts between your professional activity and legal or other accepted public requirements;
- c) advise your stakeholders as soon as possible of any conflicts of interest or conscientious objections that you have;
- d) take into consideration the fact that your profession traverses many other professions, and has implications for other social systems and organisations;
- e) endeavour to preserve the integrity, security, continuity and utility of ICT;
- f) respect the intellectual property of others; and
- g) endeavour to preserve the confidentiality and privacy of the information of others.

1.2.2. The Enhancement of Quality of Life

The development of ICT has had a significant impact on our society and way of life. Whilst this impact has been beneficial to a very great extent, like all technologies, ICT has also had some negative effects, and will continue to do so. An ethical approach to your work will help to recognise and minimise these adverse effects. You should promote equal access to the benefits of ICT by all members of society.

In accordance with this value you will:

- a) recognise, in your work, the role that ICT can play to enhance the quality of life of people, particularly the disadvantaged or those with disabilities;
- b) protect and promote the health and safety of those affected by your work;
- c) understand, and give due regard to, the perceptions of those affected by your work; and
- d) attempt to increase the feelings of personal satisfaction, competence, and control of those affected by your work.

1.2.3. Honesty

Do not breach public trust in the profession or the specific trust of your stakeholders. Observance of utmost honesty and integrity must underlie all your professional decisions and actions. Circumstances will undoubtedly arise during the course of your professional career where it may appear to be beneficial for you to be deceptive in some way. This type of behaviour is not acceptable professional conduct.

In accordance with this value you will:

- a) reject, and will not make, any offer of bribery or inducement;
- b) not knowingly mislead a client or potential client as to the suitability of a product or service;
- c) distinguish between your professional and personal opinions and advice;
- d) give realistic estimates for projects under your control;
- e) qualify professional opinions which you know are based on limited knowledge or experience;
- f) give credit for work done by others where credit is due; and
- g) not attempt to enhance your own reputation at the expense of another person's reputation.

1.2.4. Competence

Accept only such work as you believe you are competent to perform, and do not hesitate to obtain additional expertise from appropriately qualified individuals where advisable. You should always be aware of your own limitations and not knowingly imply that you have competence you do not possess. This is distinct from accepting a task of which the successful completion requires expertise additional to your own. You cannot possibly be knowledgeable on all facets of ICT but you should be able to recognise when you need additional expertise and information.

In accordance with this value you will:

- a) endeavour to provide products and services which match the operational and financial needs of your stakeholders;
- b) not misrepresent your skills or knowledge;
- c) make yourself aware of relevant standards and legislation, and act accordingly;
- d) respect and protect your stakeholders' proprietary interests;
- e) advise your stakeholders when you believe a proposed project, product or service is not in their best interest;
- f) accept responsibility for your work; and
- g) respect, and seek when necessary, the professional expertise of colleagues in their areas of competence.

1.2.5. Professional Development

Keep yourself informed of such new technologies, practices and standards as are relevant to your work. Others will expect you to provide special skills and advice; and in order to do so, you must keep your knowledge up-to-date. You should encourage your staff and colleagues to do the same. Take action to ensure that your hard-won knowledge and experience are passed on in such a way that the recipients not only improve their own effectiveness in their present work but also become keen to advance their capabilities and take on additional responsibilities.

In accordance with this value you will:

- a) continue to upgrade your knowledge and skills;
- b) increase your awareness of issues affecting the profession and its relationship with the public;
- c) encourage your colleagues, employees and students to continue their own professional development; and
- d) support education, training and professional development in ICT that reflects the diverse needs of individual professionals and their various career paths.

1.2.6. Professionalism

The ICT industry is relatively new and characterised by rapid change. It has not had the opportunity to evolve over many years and acquire its own standards and legislation. The ACS is endeavouring to improve public confidence in the ICT industry. It is imperative that members of the Society maintain professional standards that improve and enhance the industry's image, especially in the workplace.

All people have a right to be treated with dignity and respect. Discrimination is unprofessional behaviour, as is any form of harassment. Members should be aware that the ACS can help them resolve ethical dilemmas. It can also provide support for taking appropriate action, including whistle-blowing, if you discover an ACS member engaging in unethical behaviour;

In accordance with this value you will:

- a) take a calm, objective, informed and knowledgeable stance on your professional work, complementing your enthusiasm and engagement in it;
- b) take appropriate action against members who engage in behaviours contrary to this Code;
- c) confront attempts to limit diversity in the workplace, and ensure that opportunities for employment, advancement, remuneration and other working conditions are based on the actual skills and performance of employees, free of stereotypes and prejudices;
- d) note that the corporate actions of the Society are subject to this Code, and you should do whatever you can to ensure that the ACS and its officers meet this obligation;
- e) neither require, nor attempt to influence, any person to take any action which would involve a breach of this Code;
- f) refrain from any conduct or action in your professional role which may tarnish the image of the profession or detract from the good name of the ACS;
- g) endeavour to extend public knowledge and understanding of ICT;
- h) co-operate in advancing ICT by communication with other professionals, students and the public; and
- i) have pride in your profession, and protect and promote professionalism in ICT

Software Engineering Code of Ethics and Professional Practice

Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices and jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering.

- Short Version
- Full Version

Software Engineering Code of Ethics and Professional Practice (Short Version)

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Software Engineering Code of Ethics and Professional Practice (Full Version)

PREAMBLE

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code be used in isolation to justify errors of omission or commission. The list of Principles and Clauses is not exhaustive. The Clauses should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations. The Code is not a simple ethical algorithm that generates ethical decisions. In some situations standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the circumstances.

Ethical tensions can best be addressed by thoughtful consideration of fundamental principles, rather than blind reliance on detailed regulations. These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the "Public Interest" is central to this Code.

The dynamic and demanding context of software engineering requires a code that is adaptable and relevant to new situations as they occur. However, even in this generality, the Code provides support for software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions that are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. As this Code expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

PRINCIPLES

Principle 1: PUBLIC

Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

- 1.01. Accept full responsibility for their own work.**
- 1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.**
- 1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.**
- 1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.**
- 1.05. Cooperate in efforts to address matters of grave public concern caused by software, its installation, maintenance, support or documentation.**
- 1.06. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.**
- 1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that can diminish access to the benefits of software.**
- 1.08. Be encouraged to volunteer professional skills to good causes and contribute to public education concerning the discipline.**

Principle 2: CLIENT AND EMPLOYER

Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. In particular, software engineers shall, as appropriate:

- 2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.**
- 2.02. Not knowingly use software that is obtained or retained either illegally or unethically.**
- 2.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.**
- 2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.**
- 2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.**
- 2.06. Identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.**
- 2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client.**
- 2.08. Accept no outside work detrimental to the work they perform for their primary employer.**
- 2.09. Promote no interest adverse to their employer or client, unless a higher ethical**

concern is being compromised; in that case, inform the employer or another appropriate authority of the ethical concern.

Principle 3: PRODUCT

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.

3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.

3.04. Ensure that they are qualified for any project on which they work or propose to work by an appropriate combination of education and training, and experience.

3.05. Ensure an appropriate method is used for any project on which they work or propose to work.

3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.

3.07. Strive to fully understand the specifications for software on which they work.

3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements and have the appropriate approvals.

3.09. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work and provide an uncertainty assessment of these estimates.

3.10. Ensure adequate testing, debugging, and review of software and related documents on which they work.

3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.

3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.

3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.

3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.

3.15 Treat all forms of software maintenance with the same professionalism as new development.

Principle 4: JUDGMENT

Software engineers shall maintain integrity and independence in their professional

judgment. In particular, software engineers shall, as appropriate:

4.01. Temper all technical judgments by the need to support and maintain human values.

4.02 Only endorse documents either prepared under their supervision or within their areas of competence and with which they are in agreement.

4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.

4.04. Not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.

4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.

4.06. Refuse to participate, as members or advisors, in a private, governmental or professional body concerned with software related issues, in which they, their employers or their clients have undisclosed potential conflicts of interest.

Principle 5: MANAGEMENT

Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance . In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.

5.02. Ensure that software engineers are informed of standards before being held to them.

5.03. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files and information that is confidential to the employer or confidential to others.

5.04. Assign work only after taking into account appropriate contributions of education and experience tempered with a desire to further that education and experience.

5.05. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates.

5.06. Attract potential software engineers only by full and accurate description of the conditions of employment.

5.07. Offer fair and just remuneration.

5.08. Not unjustly prevent someone from taking a position for which that person is suitably qualified.

5.09. Ensure that there is a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed.

5.10. Provide for due process in hearing charges of violation of an employer's policy or of this Code.

5.11. Not ask a software engineer to do anything inconsistent with this Code.

5.12. Not punish anyone for expressing ethical concerns about a project.

Principle 6: PROFESSION

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

6.01. Help develop an organizational environment favorable to acting ethically.

6.02. Promote public knowledge of software engineering.

6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.

6.04. Support, as members of a profession, other software engineers striving to follow this Code.

6.05. Not promote their own interest at the expense of the profession, client or employer.

6.06. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.

6.07. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful.

6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.

6.09. Ensure that clients, employers, and supervisors know of the software engineer's commitment to this Code of ethics, and the subsequent ramifications of such commitment.

6.10. Avoid associations with businesses and organizations which are in conflict with this code.

6.11. Recognize that violations of this Code are inconsistent with being a professional software engineer.

6.12. Express concerns to the people involved when significant violations of this Code are detected unless this is impossible, counter-productive, or dangerous.

6.13. Report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counter-productive or dangerous.

Principle 7: COLLEAGUES

Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:

7.01. Encourage colleagues to adhere to this Code.

7.02. Assist colleagues in professional development.

- 7.03. Credit fully the work of others and refrain from taking undue credit.**
- 7.04. Review the work of others in an objective, candid, and properly-documented way.**
- 7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.**
- 7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures for protecting passwords, files and other confidential information, and security measures in general.**
- 7.07. Not unfairly intervene in the career of any colleague; however, concern for the employer, the client or public interest may compel software engineers, in good faith, to question the competence of a colleague.**
- 7.08. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.**

Principle 8: SELF

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. In particular, software engineers shall continually endeavor to:

- 8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance and testing of software and related documents, together with the management of the development process.**
- 8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.**
- 8.03. Improve their ability to produce accurate, informative, and well-written documentation.**
- 8.04. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.**
- 8.05. Improve their knowledge of relevant standards and the law governing the software and related documents on which they work.**
- 8.06 Improve their knowledge of this Code, its interpretation, and its application to their work.**
- 8.07 Not give unfair treatment to anyone because of any irrelevant prejudices.**
- 8.08. Not influence others to undertake any action that involves a breach of this Code.**
- 8.09. Recognize that personal violations of this Code are inconsistent with being a professional software engineer.**

This Code was developed by the ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices (SEEPP):

Executive Committee: Donald Gotterbarn (Chair), Keith Miller and Simon Rogerson;

Members: Steve Barber, Peter Barnes, Ilene Burnstein, Michael Davis, Amr El-Kadi, N. Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Mark Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manuel J. Norman, Douglas Phillips,

Peter Ron Prinzivalli, Patrick Sullivan, John Weckert, Vivian Weil, S. Weisband and Laurie Honour Werth.

This Code may be published without permission as long as it is not changed in any way and it carries the copyright notice. Copyright (c) 1999 by the Association for Computing Machinery, Inc. and the Institute for Electrical and Electronics Engineers, Inc.

SWEN90016: Software Processes and Management
Ethical Case Studies

Case Study 1: AUSTRALIAN COMPUTER SOCIETY CODE OF ETHICS – CASE STUDIES

Tax Software Package

A software development company has just produced a new software package that incorporates the new tax laws and figures taxes for both individuals and small businesses.

The president of the company knows that the program has a number of bugs. He also believes the first firm to put this kind of software on the market is likely to capture the largest market share. The company widely advertises the program. When the company actually ships a CD, it includes a disclaimer of responsibility for errors resulting from the use of the program. The company expects it will receive a number of complaints, queries, and suggestions for modification. The company plans to use these to make changes and eventually issue updated, improved, and debugged versions. The president argues that this is general industry policy and that anyone who buys version 1.0 of a program knows this and will take proper precautions.

Because of bugs, a number of users filed incorrect tax returns and were penalized by the ATO.

Source: http://www.acs.org.au/publication/docs/ACS_CaseStudiesFinal.pdf and Burmeister, O.K, (2000), Applying the ACS Code of Ethics, Journal of Research & Practice in Information Technology, Vol. 32, No. 2, May

Questions

1. What would you have done?
2. How could the ACS code of ethics have guided you?