THE UNIVERSITY OF MELBOURNE

SWEN90016

Software Processes & Project Management

Introduction

Project Initiation

Medic Case Study

Assignment 1

2021 – Semester 1

Tutorial 1

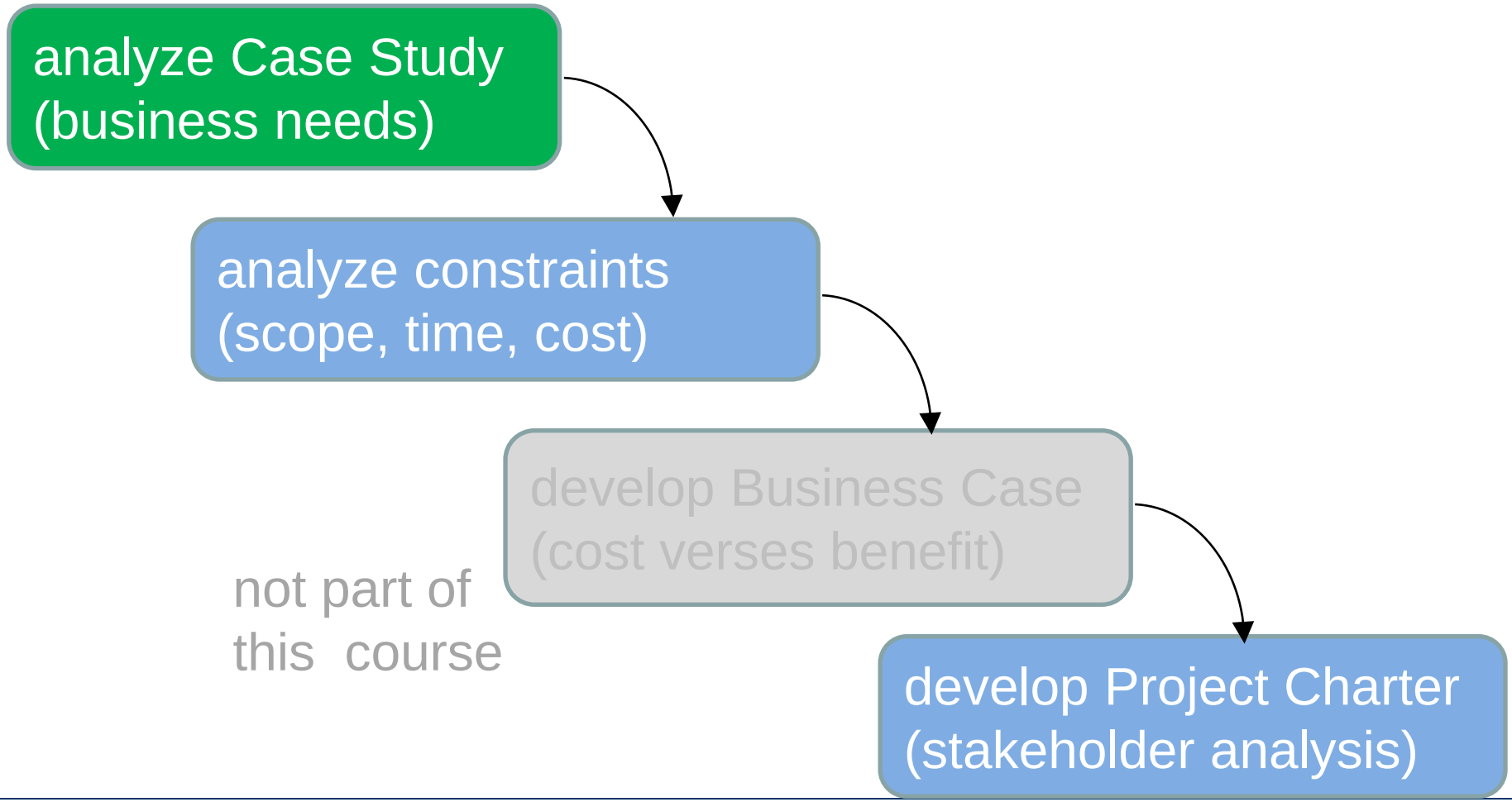
Today's aim

Understand the initialization phase by doing an activity for each phase

1. Business needs analysis
2. Analyse constraints
3. Stakeholder analysis

Initialization Phase

The first Project Management ***process***: initialization



Project Management Terminology

Activity: You want to cycle from Melbourne to Sydney.

Groups of 4

analyze Case Study (business needs)

- What are the challenges for such a project?
- What risks would this project need to consider?

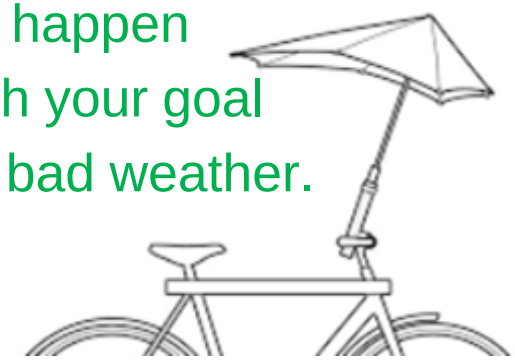
Project Characteristics

Project: You want to cycle from Melbourne to Sydney.

Goal:
- Create a fun & exciting adventure

Challenge: It is hard to travel this long distance with a bicycle
- This characteristic is known to exist.
- The solution requires resources, (fitness).

Risk: The weather may be very bad (cold or rain)
- This possible future event may or may not happen
- Bad weather may cause you to fail to reach your goal
- Better plan a mitigation strategy to control bad weather.

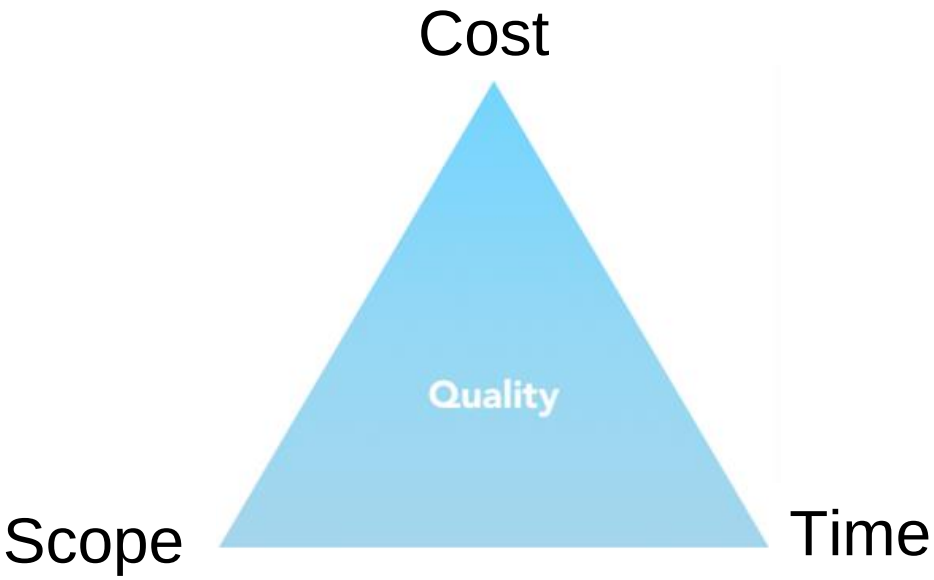


Project Constraints

Know your project's

triple constraint

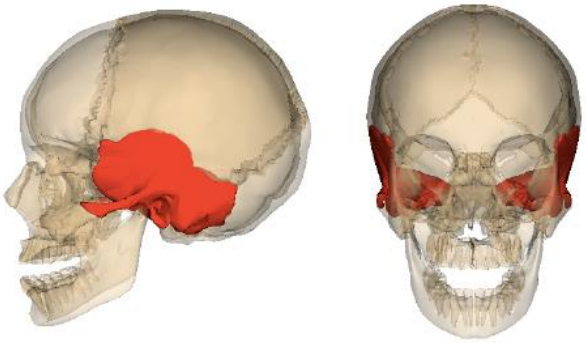
analyze constraints (scope, time, cost)



Medic Case Study

Case Study 1 -Virtual Temporal Bone Surgery

- Who has read the Case Study?
- Do you know what Person Days are?
- Divide the Case Study into components



What kind of system is this?

- What are the project's characteristics?

Medic Case Study

Discuss and plan and question

Get into groups of 4-5 people.

Fill in the exercise sheet.

Person Days clues:

- Project duration is 2 university semesters = 30 weeks
- 4th year SWEN students take 4 subjects a semester = ¼ time allocation
- IT support developers at $50,000 pa = fractional time allocation

Cost clues:

- experienced developers at $100,000 pa
- experienced surgeons at $200,000 pa
- junior developers (4th year SWEN students) at zero cost
- junior surgeon users at zero cost
- IT support developers at $50,000 pa

Medic Case Study Exercise

Project Information and Estimation

Item	Value	Reason
Team Size		
Person Days		
Cost		
Project Goal		
Key Characteristics	VR simulation enacting real-world tasks, integrate multiple hardware devices, good graphics & embedded software, QA focus	
Possible Risks		

Project Risks

Know your project's *risks*

- If this project was to fail, what do you think would be the reason?

- What harm minimization strategies would you plan to use?

Risks

1. Unexpected behavior of new hardware
2. Low technical ICT skills of users
3. Skill set of developers in C++
4. Compare availability of Graphical libraries in C++ / Java
5. Algorithmic complexity

Thank You!

SWEN90016

Software Processes & Project Management

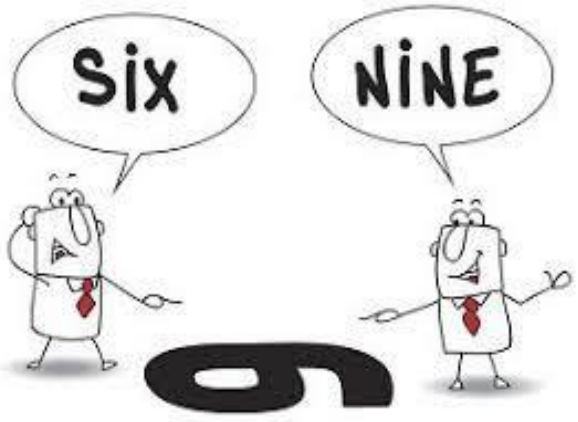
Stakeholder analysis and Communication

2021 – Semester 1

Tutorial 2

Today's aim

Stakeholder Analysis
Communication




Project Charter, Scope
Project Schedule
Project Resource
Project Budget & Cost
Project Quality
Project Risk
Project Communication
Project Planning

Project Stakeholders

Know your stakeholders

Who is invested?

develop Project Charter (stakeholder analysis)

- Prioritize and understand your stakeholders
- Someone's position on the grid shows you the actions you have to take with them
- How do they feel about the project

Project Stakeholders

Know your stakeholders

Who is invested?

develop Project Charter (stakeholder analysis)

- Discuss in groups

- The University of Melbourne
- A new timetabling system
- Who do you think would be the stakeholders
- Project champion

Stakeholder Communications

What should you tell them?

Communication plan (PMP)

Determine how you communicate:

You are the Project Manager and the project is experiencing delays as one of the components is not working.

Discuss how you will inform each of the stakeholders.

Project Stakeholders

Know your stakeholders

Who is invested?

develop Project Charter (stakeholder analysis)

Hi power/low interest: Sponsors, such as C-level Executive, Head of Faculty at UniMelb, who allocates funds to 101 different projects

Hi power/hi interest: such as upper-level Executive, who's reputation is invested in outcome

Project Stakeholders

Know your stakeholders

Who is invested?

develop Project Charter (stakeholder analysis)

Low power/low interest: Support team, who routinely manage other projects

Low power/hi interest: Project team, who would otherwise need to find another job

Champion is hi influence role who advocates the benefits of the project.

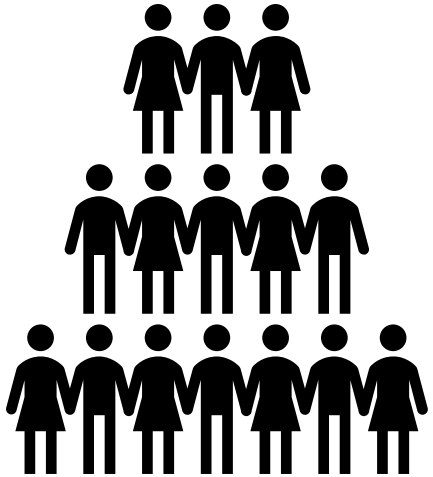
Today's aim

Explore Communication

Virtual Teams: Communication

Modes of Communication

- Skype
- Zoom
- WhatsApp
- WeChat
- Google drive



Frequency of Communication

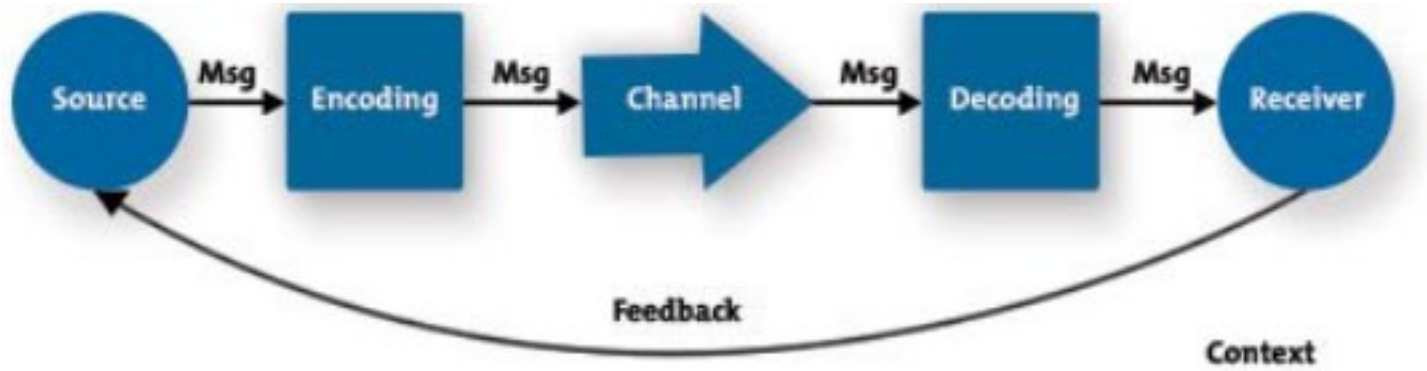
- Daily
- Weekly
- Monthly

Communication Models

What is your own experience of communication?

Draw your own model

- Who
  - Says What
    - How
      - To Whom
        - Outcome



Interpersonal Communication

Exercise- Role play

What if you were the manager?

What if you were a member of the team?

Alex is a brilliant programmer

Alex has been late 3 of the last 10 days

Alex missed a meeting

Alex got the specifications wrong and wasted 2 days of coding time because of a missed meeting

Has been at the company for 3 years and is usually on time

Alex does not eat lunch with the other people in the company

S.M.A.R.T. Feedback

In your groups, **discuss** an appropriate feedback message.

Hi there Alex,

You're usually one of the first employees in the meeting room and I've noticed recently that there's been a few occasions where you've been late or missed a meeting. If this happens in the future could you, please follow up on the meeting minutes just to ensure that you've kept up to date regarding project specifications.

Specific Measurable Achievable Relevant Timely

Interpersonal Communication

Skills • Listening

1. Direct probe
2. Open
3. Closed
4. Objective criteria
   4. facts to defuse contention
5. Testing
   5. already know the answer
6. Softening up
   6. build rapport
7. Hypothetical
   7. "what if" exploration
8. Reflective
   8. acknowledge feeling
9. Leading
   9. suggests one answer
10. Rhetorical
11. Stupid
12. Trick compliment
13. Back on Track
14. False dilemma

What, why and when is an OPEN question appropriate
What, why and when is is a CLOSED question appropriate

Thank You!

SWEN90016

Software Processes & Project Management

SDLC Process
Language Research Case Study
Groups- Assignment 2

2021– Semester 1

Tutorial 3

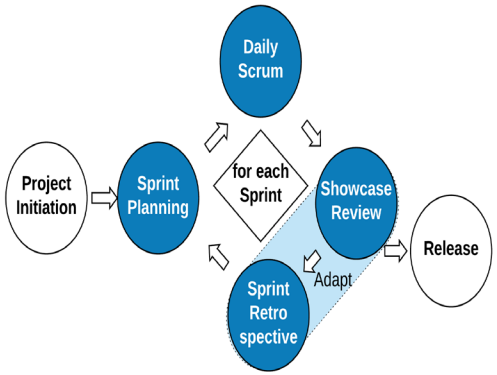
Case Study Analysis

Case Study 2 –Language Research Tool Project

In groups of 4- Determine the goals & key characteristics

Goal

- Efficient, open & published language learning
- Cultural preservation



Key Characteristics

- Data centric, data integrity, data sharing
- Complex privileges & roles
- Global, distributed, interoperable

Today's aim

Understand the differences between the SDLC and when to use which one

Waterfall

V-model

Incremental

Agile

Project Analysis

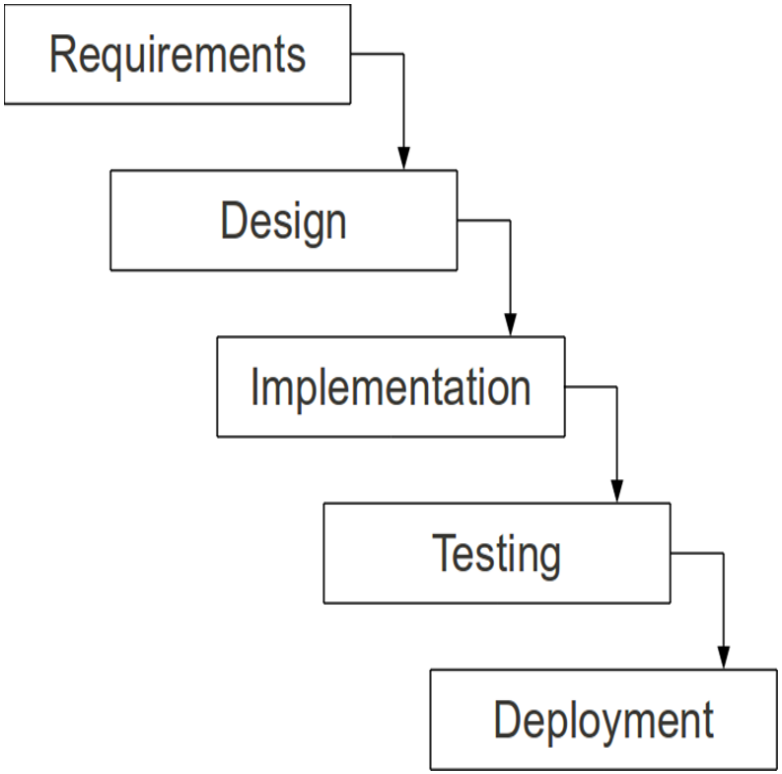
Get into 2 groups, **discuss** SDLC processes & activities

1) Consider the project, people & technology
   1) Evaluate the team and project constraints.

2) Evaluate multiple SDLC models
   1) Identify the advantages and disadvantages
   2) Consider the outcomes and risks

3) Justify an effective SDLC process
   1) Defend your choice of SDLC
   2) List the activities done
      - Formal: Requirements Specification
      - Agile: Product Backlog

Waterfall



What are the Advantages?

- Simple and easy management
- Rigid and sequential
- Documentation produced
- Requirements stable and precise

What are the Disadvantages?

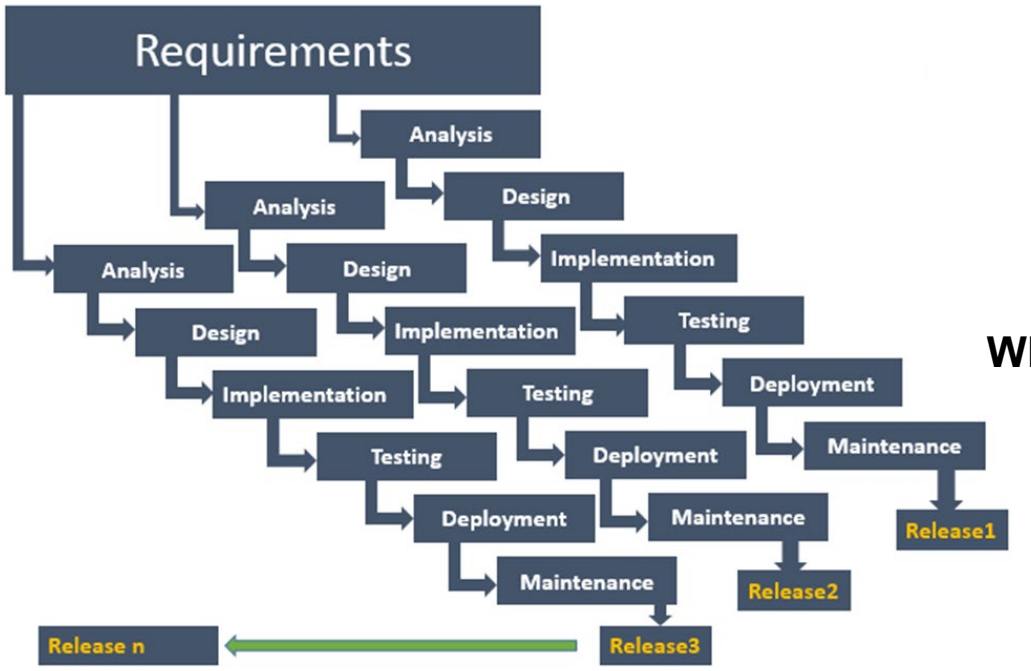
- Bad news known late in process
- Client feedback known late in process
- Discourages change
- Documentation not valued
- Risks and changes have big impact

Incremental

Requirement - partition into segments

Releases - mini waterfall process

Integrate modules



What are the Advantages?

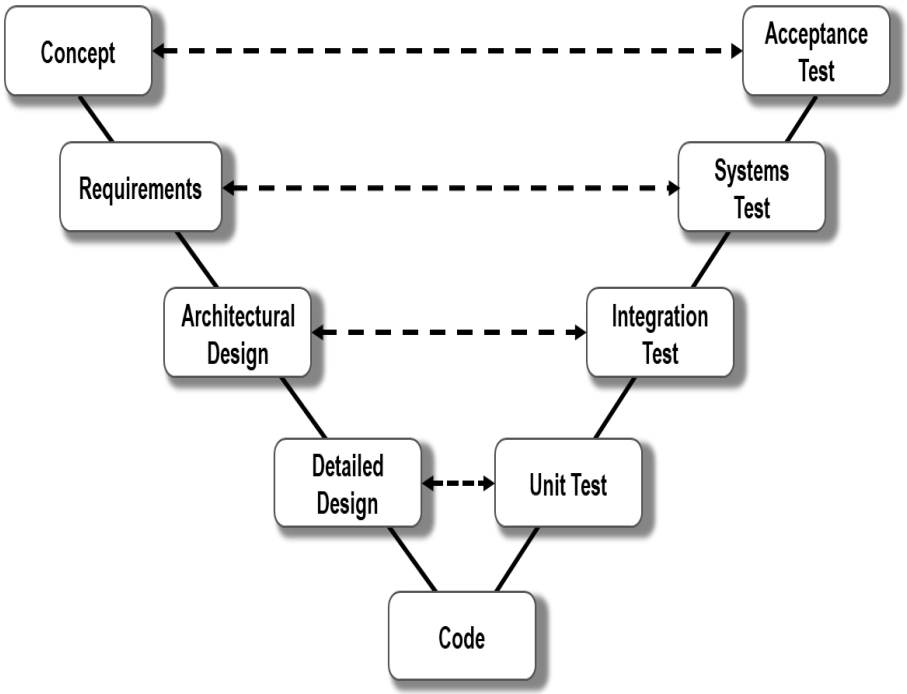
- Smaller, easier modules
- Initial modules released earlier
- Client feedback known earlier
- Change has less impact
- Requirements stable and precise

What are the Disadvantages?

- Management complexity
- Increased cost
- Partition skill
- Integration risk
- Rigid within each partition

V-Model

waterfall model plus defined artifact deliverable
development stage ← → testing phase



What are the Advantages?

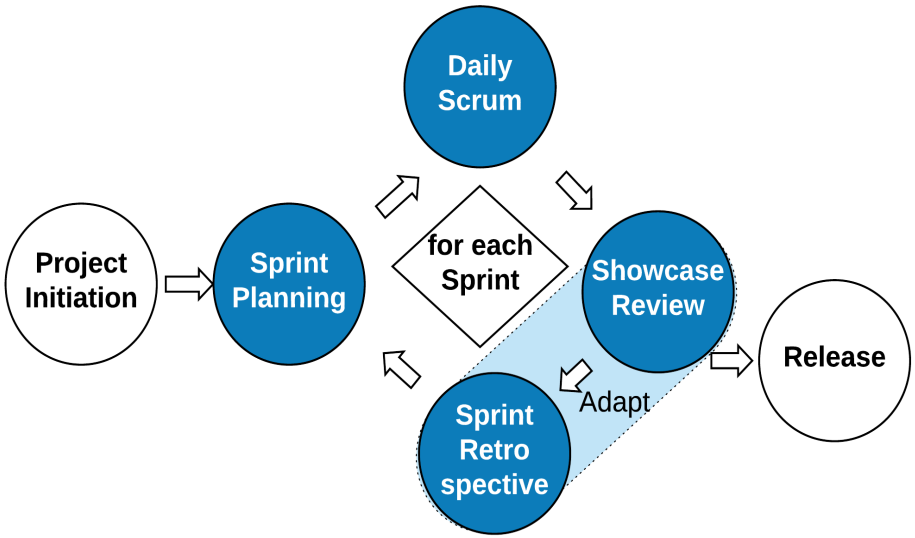
- Simple and easy management
- Rigid and sequential deliverable
- Documentation produced
- Requirements stable and precise

What are the Disadvantages?

- Requires discipline
- Bad news known late in process
- Client feedback known late in process
- Discourages change
- Test artifacts are expensive
- Risks and changes have big impact

Agile

Scrum: method to organize working teams
XP: method to improve code quality



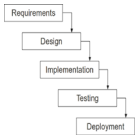
What are the Advantages?

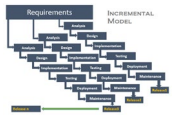
- Transparent productivity due to fast releases
- Focus on client satisfaction
  - Embraces change
  - Requirements emerge
- Efficient and simple code

What are the Disadvantages?

- Requires experience of ceremonies
- Requires teamwork skills
- Giant “TODO” list lacks design overview

Compare SDLCs

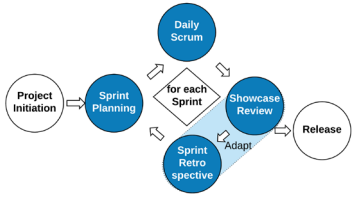





When to choose Formal Models?

- Customer knows what they want at the start
- Stable, precise and known requirements
- Change is not expected
- Mature technologies and tools

When to choose Agile Models?



- Customer gives time to project
- Requirements continue to emerge
- Change is welcome

When to choose a hybrid model?

- Client has a prescriptive model established

Case Study Analysis

Case Study 2 –Language Research Tool Project

What are the challenges and risks?

Challenges: what would make this project difficult?

- integrated, trust-worthy, searchable data

Risks: what could make this project fail?

- poor quality data and duplicate data silos

Thank You!

SWEN90016

Software Processes & Project Management

Risk Management

2021 – Semester 1
Tutorial 4

Today's aim

Understand Risk Management

for Language Research project

Recap: Challenges vs. Risks

Challenge:

- This characteristic is known to exist.
- The solution requires resources, (fitness).

Risk:

- This possible future event may or may not happen
- Better to put in place a strategy to control the event.

Risk Activities Flow

Risk Management Planning

(1 Decide how)

(2 Find them)

Risk Identification

Qualitative Risk Analysis

3-a: Evaluate probability, impact & priority

Quantitative Risk Analysis

3-b: Document

Risk Response Planning

4. Decide Strategy

Risk Monitoring & Control

5. Act & measure

.

Risk Management example

For the Language Research Case Study

Some key characteristics are:

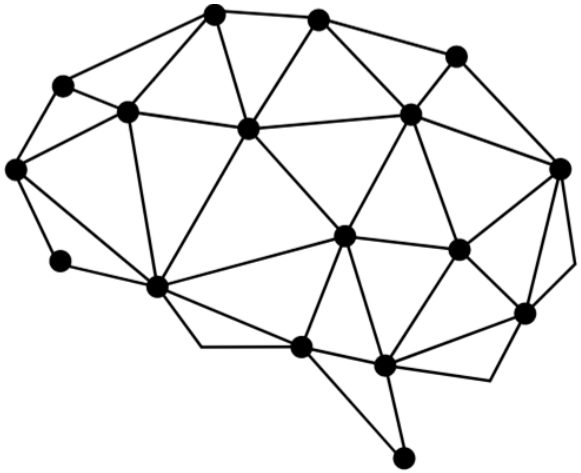
- data integration from several distributed resources in several different formats. One feature of the system is to resolve the differences between idiosyncratic data formats.

- sharing modules in a distributed environment, choose between a local installation which will generate multiple copies, or a central installation with remote access

- peer-to-peer control strategy

Analyze

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 Register: triggers & consequences

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

Risk Register: objective, owner, benchmark

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	R7	Failure to deliver	Chief Executive Assistant Director	1. Monitoring by Safer Communities Group	M	Annual external audit. Safer Communities

Information Technology

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

Analyze: Risk Register

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

Id	Risk Description
Risk 1	
Risk 2`	
Risk 3	

Analyze

5. Calculate: probability impact exposure

Risk	Probability of Risk	Size of Loss (Days)	Risk Exposure (Days)
Risk 1		*the impact to the schedule if the risk did occur*	
Risk 2`			
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

Analyze

5. Calculate: probability impact exposure

Risk	Probability of Risk	Size of Loss (Days) Out of 30 days total For 6 week project	Risk Exposure (Days)
Risk 1 –Flat priority causes ineffective automation	10%	5 days **Moderate - 3**	0.5 days
Risk 2` Data security	5%	15 days **Severe - 4**	0.75 days
Risk 3 – dev team delay	25%	2 days **Minimal - 2**	0.5 days

Respond, Monitor & Control

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

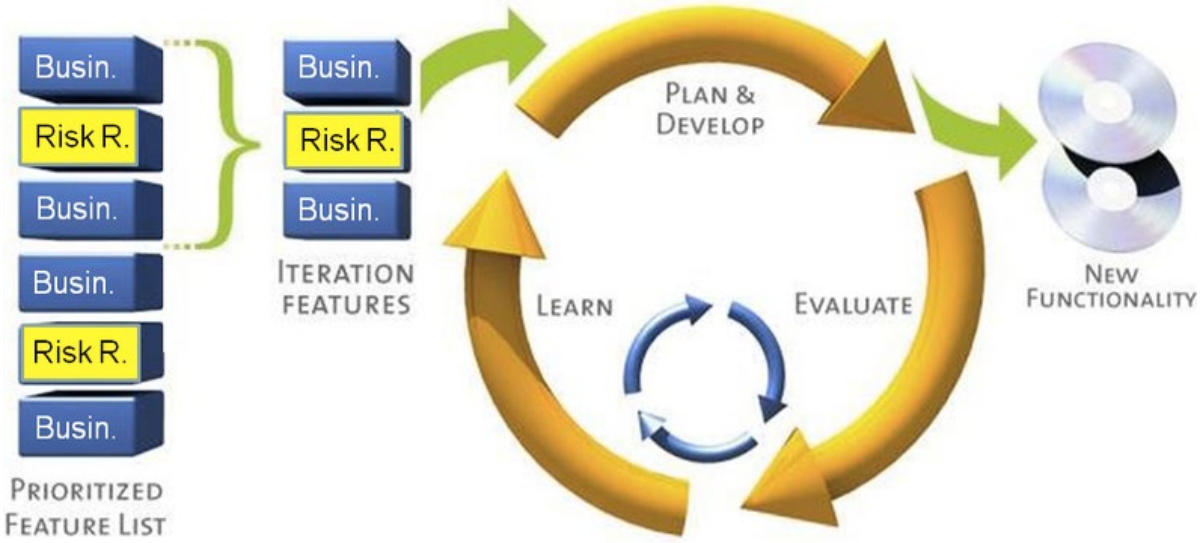
Status meetings

Agile SDLC Risk Management

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

Agile Risk Process

Sprint Review risk evaluation

What
Where
Who
When
Why

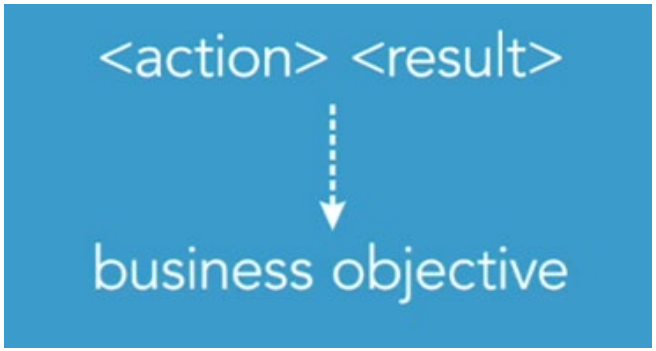
- Build small piece of working software with minimal features
- Showcase the product chunk to the stakeholders **early**
- Fail **fast** and as cheaply as possible, & get timely feedback
- Capture the ***risk item*** in the Product Backlog
- The Product Owner sets the priority of the ***risk item***

Agile Risk Process

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

Control

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

Formal	Agile
Explicit architecture	*Productivity increase*
Explicit UX design	*Responsive to feedback*
(end user consideration)	*(client satisfaction)*
Explicit configuration	*Working software*

b. How would these characteristics influence risk management?

Plan ahead	*Plan Just-In-Time*

Risk- bad examples

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

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

Risk- good examples

There is a risk that:

- *"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!

That's interesting!

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

http://www-935.ibm.com/services/us/gbs/bus/pdf/gbe03100-usen-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

SWEN90016

Software Processes & Project Management

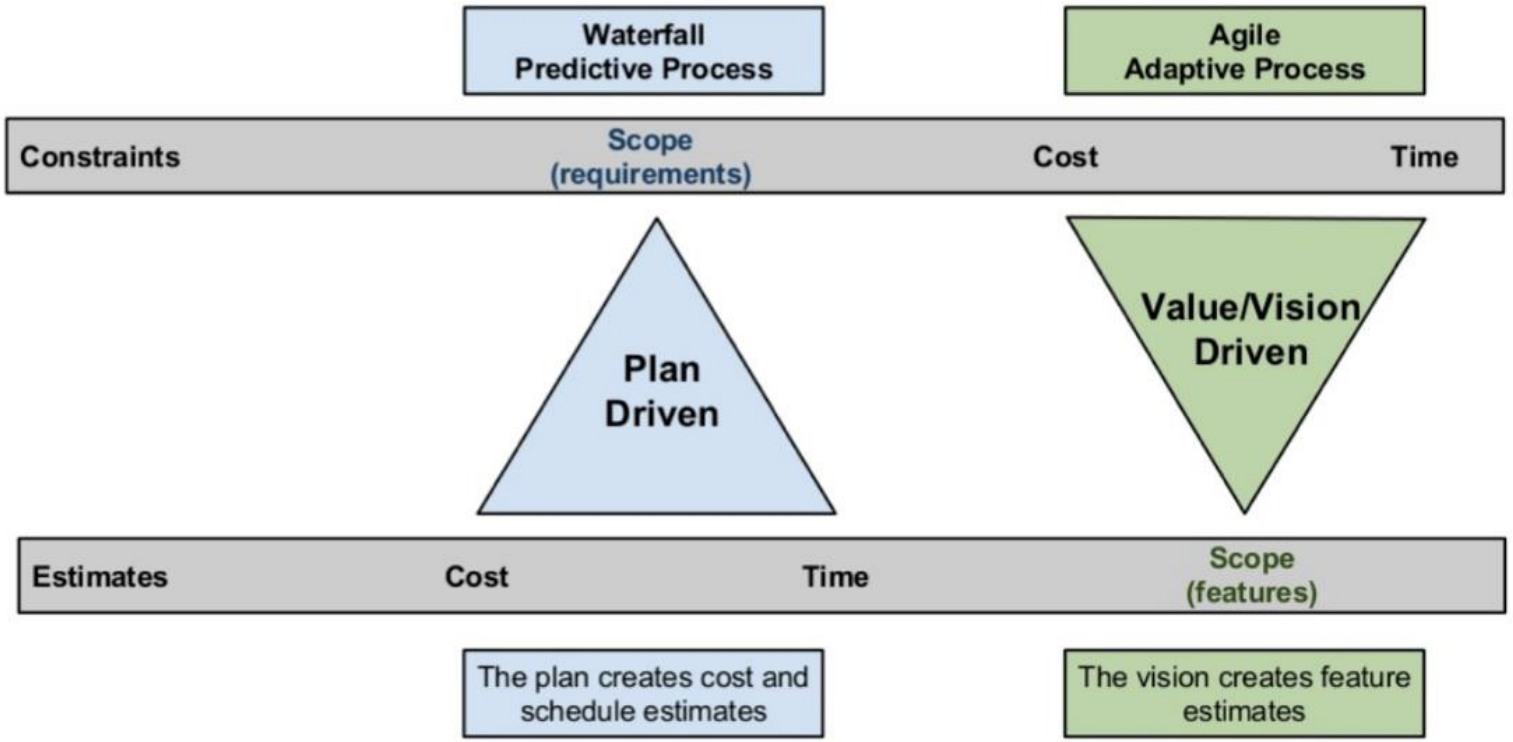
Project Planning and Scheduling

2021 – Semester 1

Tutorial 5

Today's aim

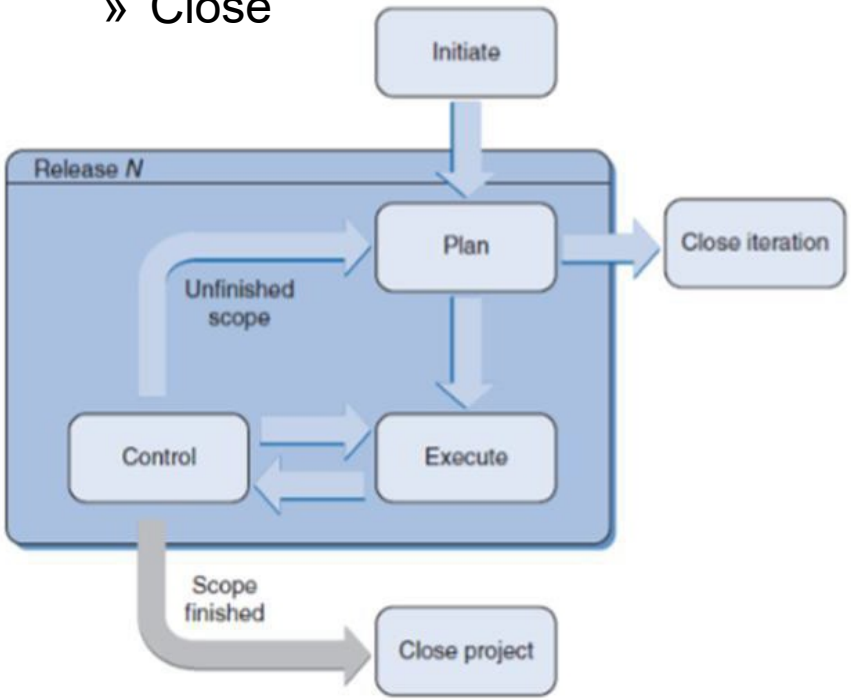
How to plan and control the schedule of software projects.



Software Projects

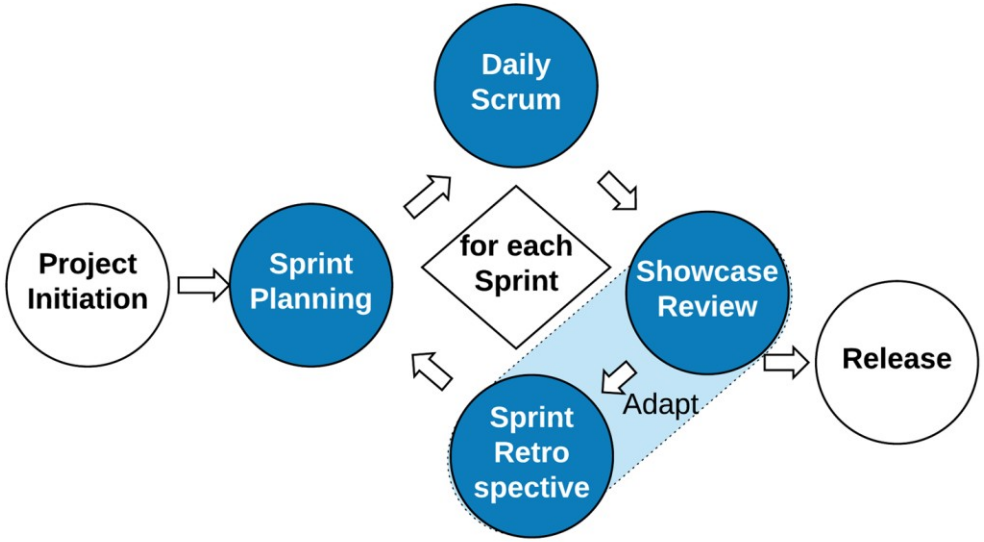
Formal PM Stages:

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



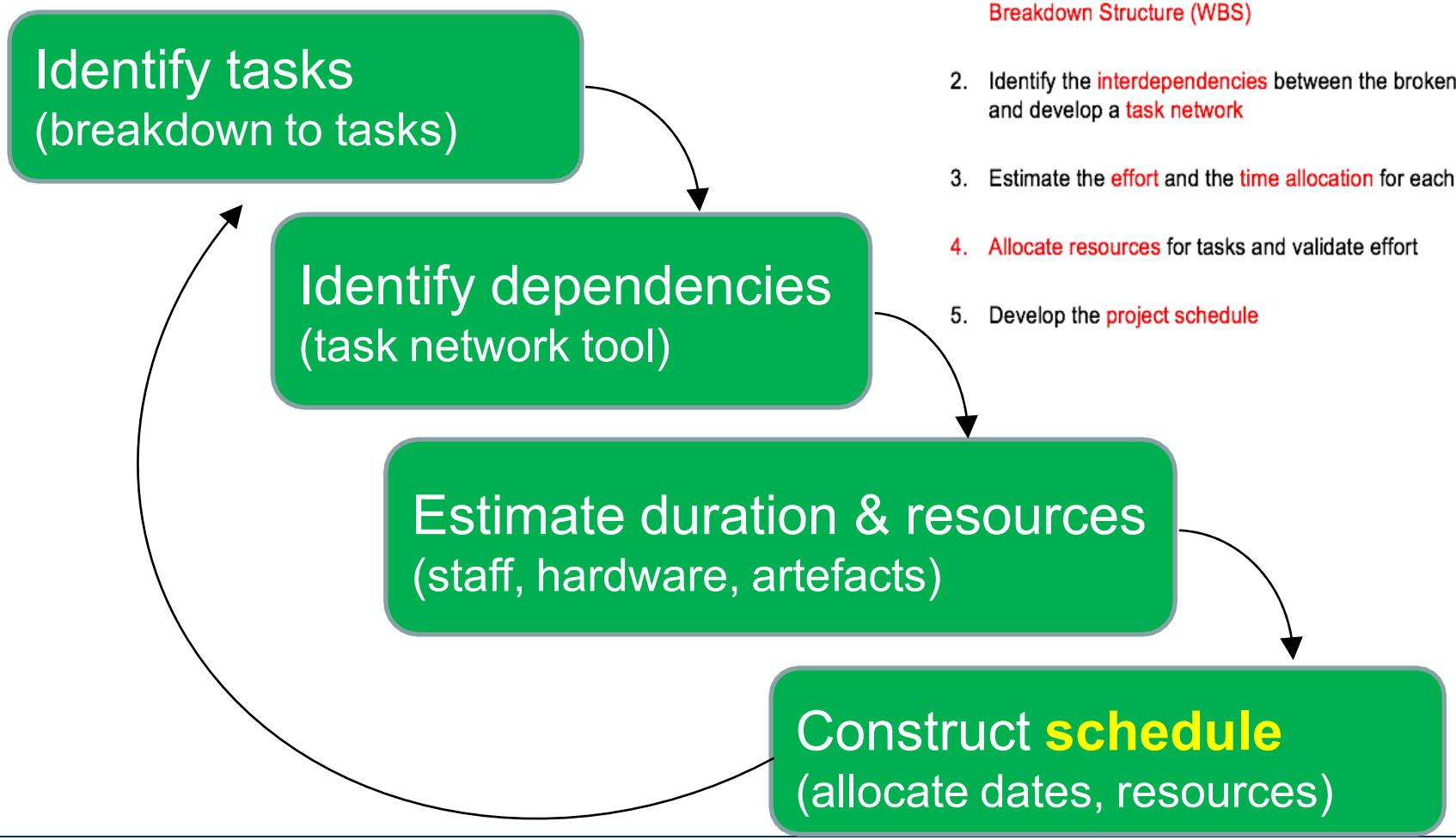
Agile PM Stages:

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



Formal Project Schedule

What steps are involved in developing a project 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 the project schedule

Formal Project Scheduling

1. Work Breakdown Structure

how to plan the schedule

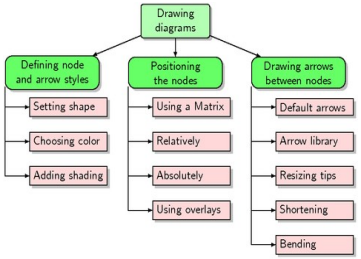
Redecorate Room

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

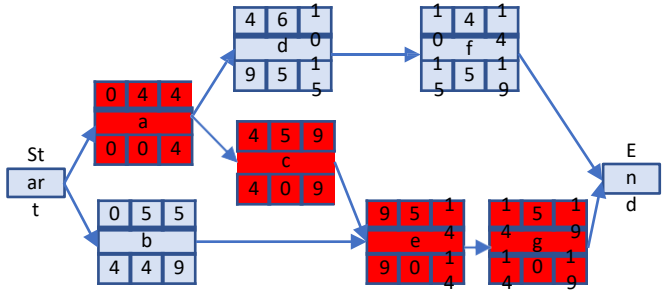
Prepare room
- Remove old wallpaper
- Remove detachable decorations
- Cover floor with old newspapers
- Cover electrical outlets/switches with tape
- Cover furniture with sheets

Paint the room

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

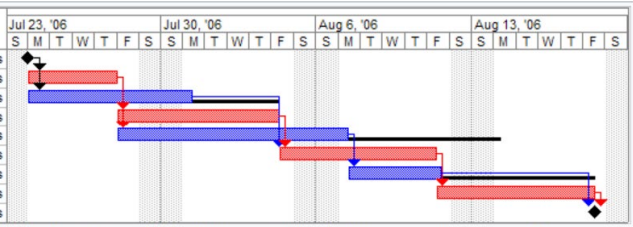


2. PERT Chart



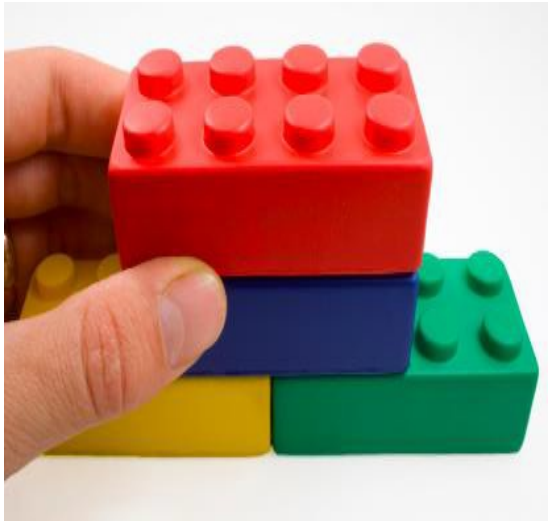
3. Gantt Chart

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



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.

Identify Tasks - Work Breakdown



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

Identify Dependencies

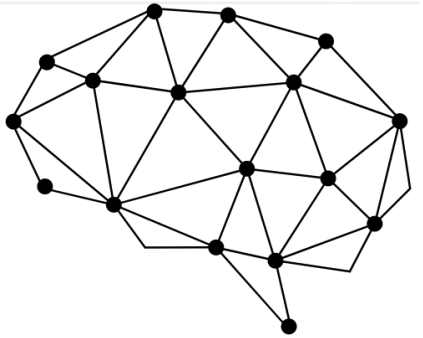
	Activity	Work Breakdown	Dependencies predecessor	Duration
1.		**Concept Phase**		
	1.1	Concept Planning		1
	1.2	Initial Research		4
	1.3	Problem definition with client		2
	1.4	Initial Project Plan	1.1, 1.2, 1.3	1
2.		**Requirements**		
	2.1	Requirements Iteration 1		
		2.1.1 Requirement Elicitation	1.4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

Identify Dependencies

Develop a task network

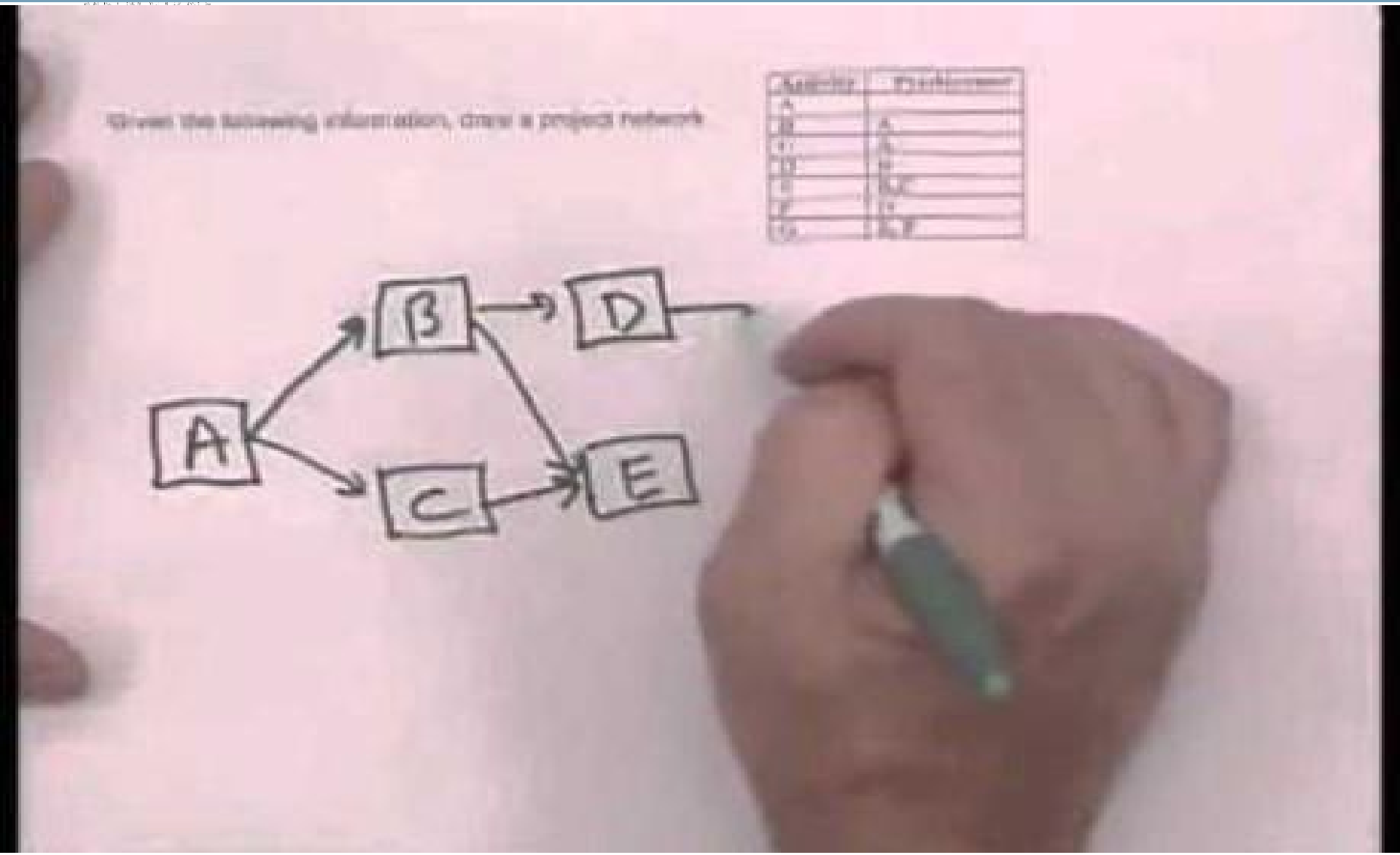
(activity on node)

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

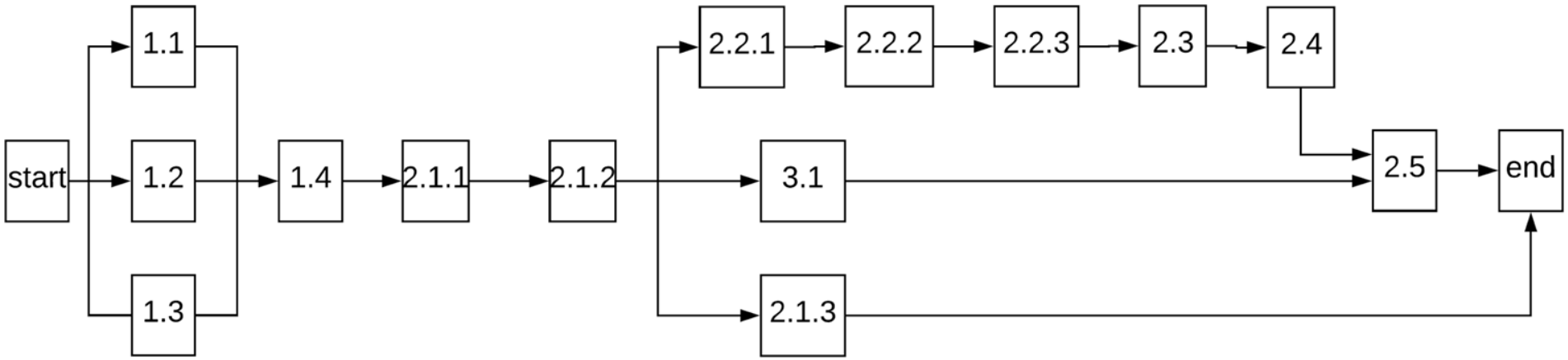
How to draw a simple network diagram



Task Network

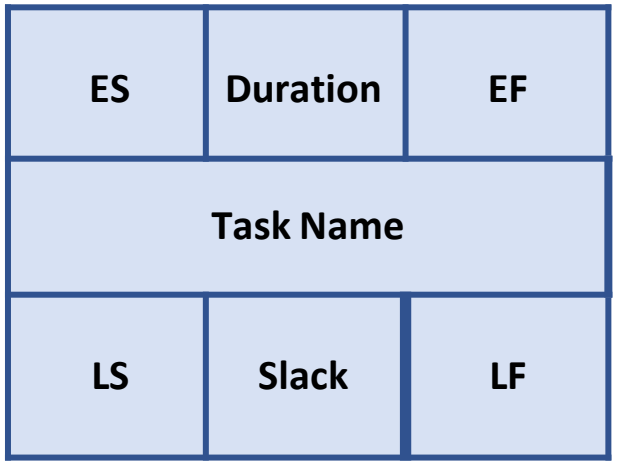
Network Diagram

- Sequential nodes
- Few details



Pert Chart

PERT: Program Evaluation & Review Technique



The activity node

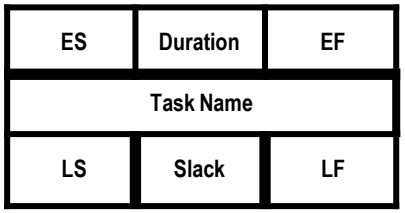
Earliest start time (ES)
Duration in people days
Earliest finish time (EF)

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

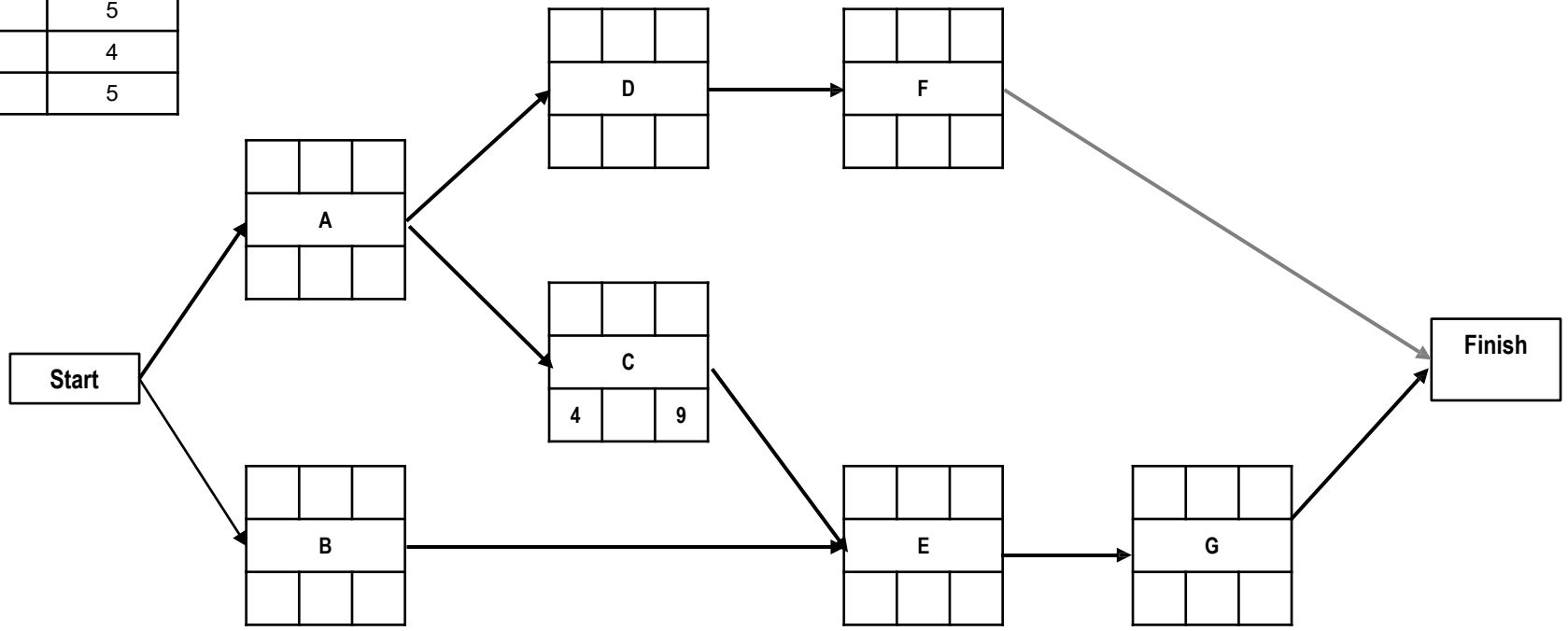
Pert Chart: example

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

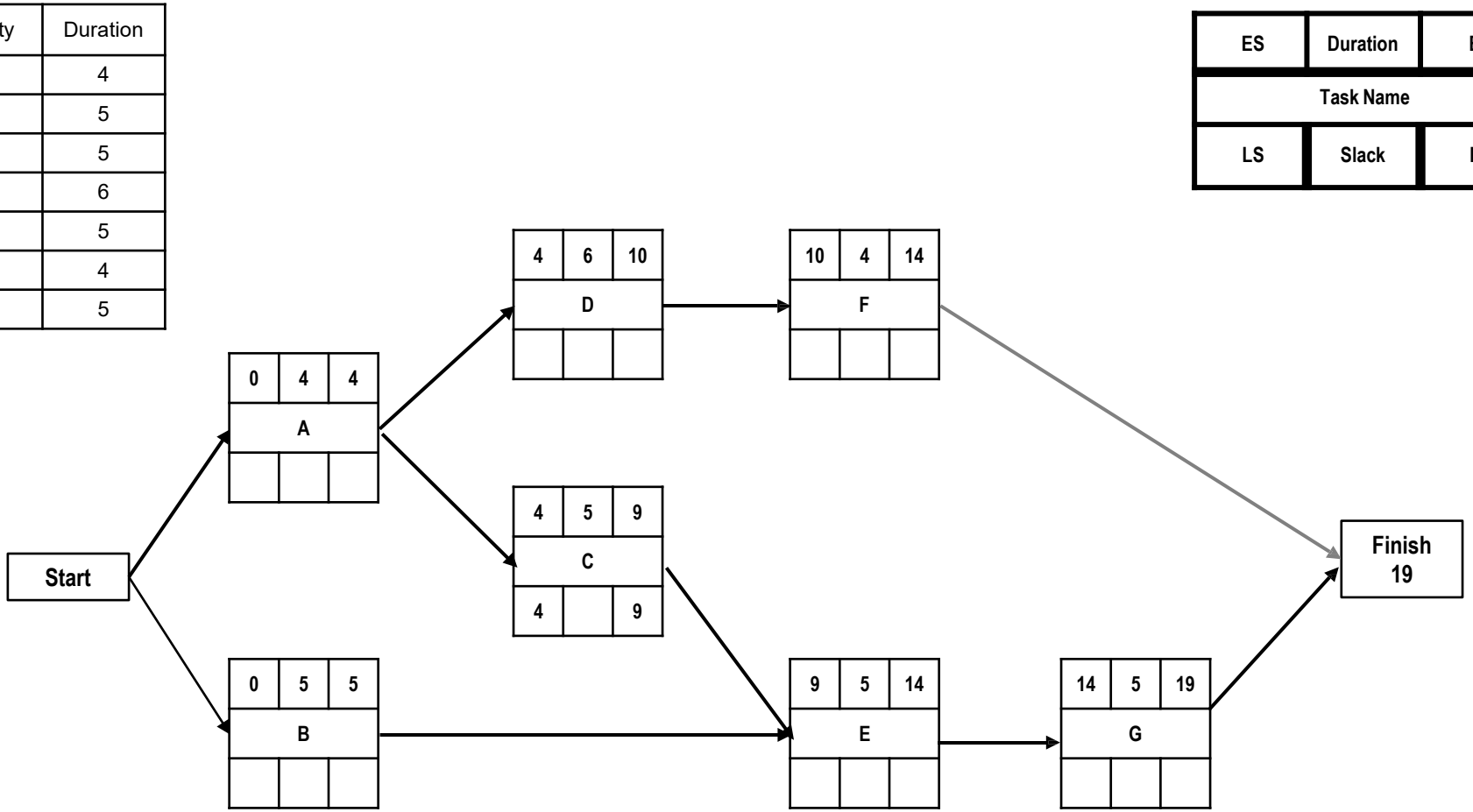


Task Network Diagram



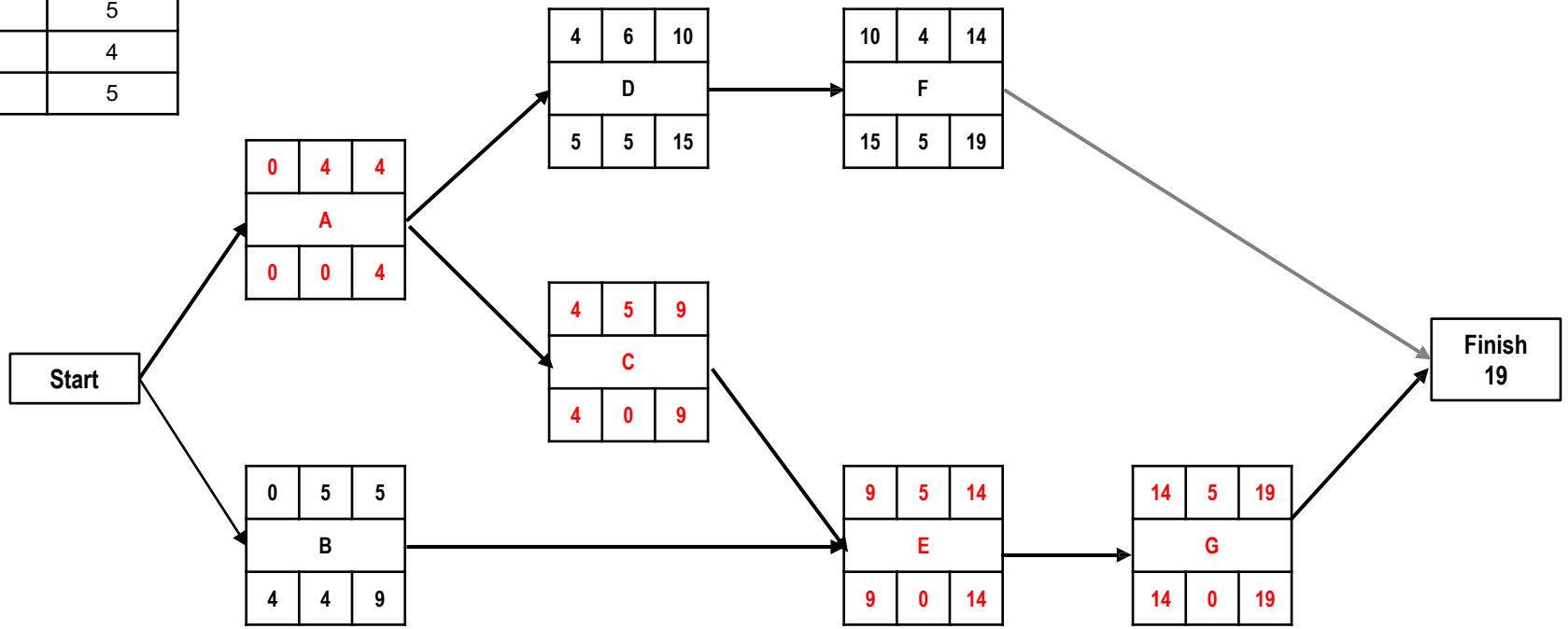
Forward Pass

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



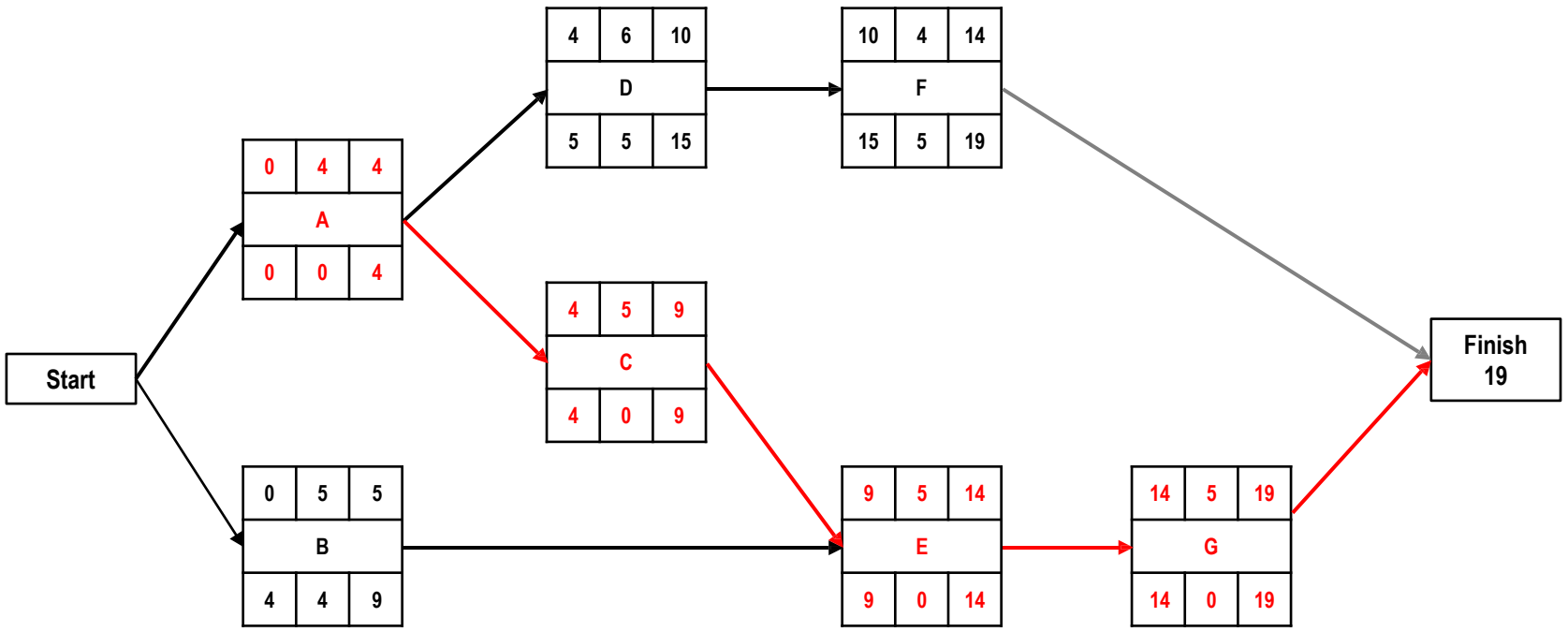
Slack

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



Critical Path

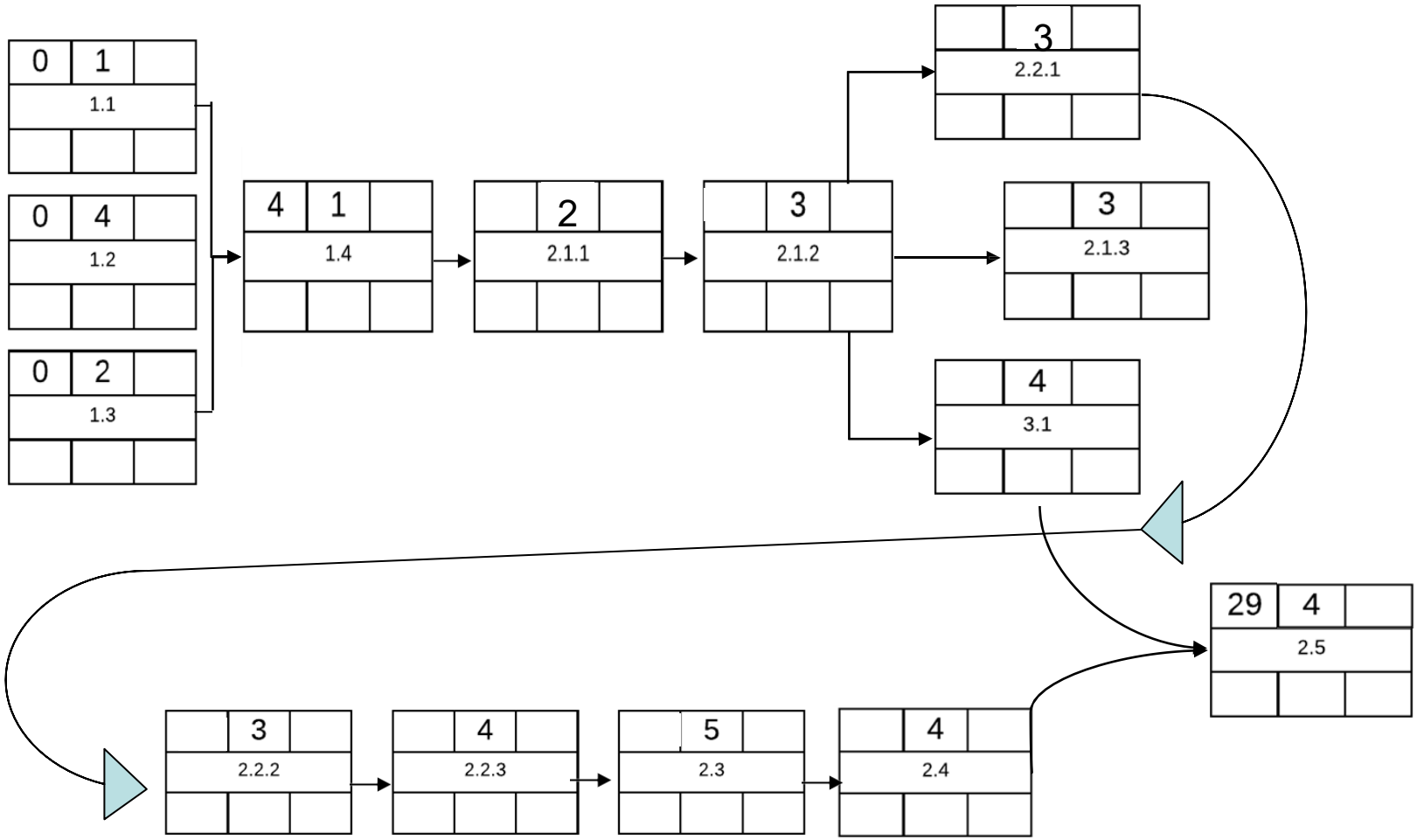
Critical Path = A + C + E + G



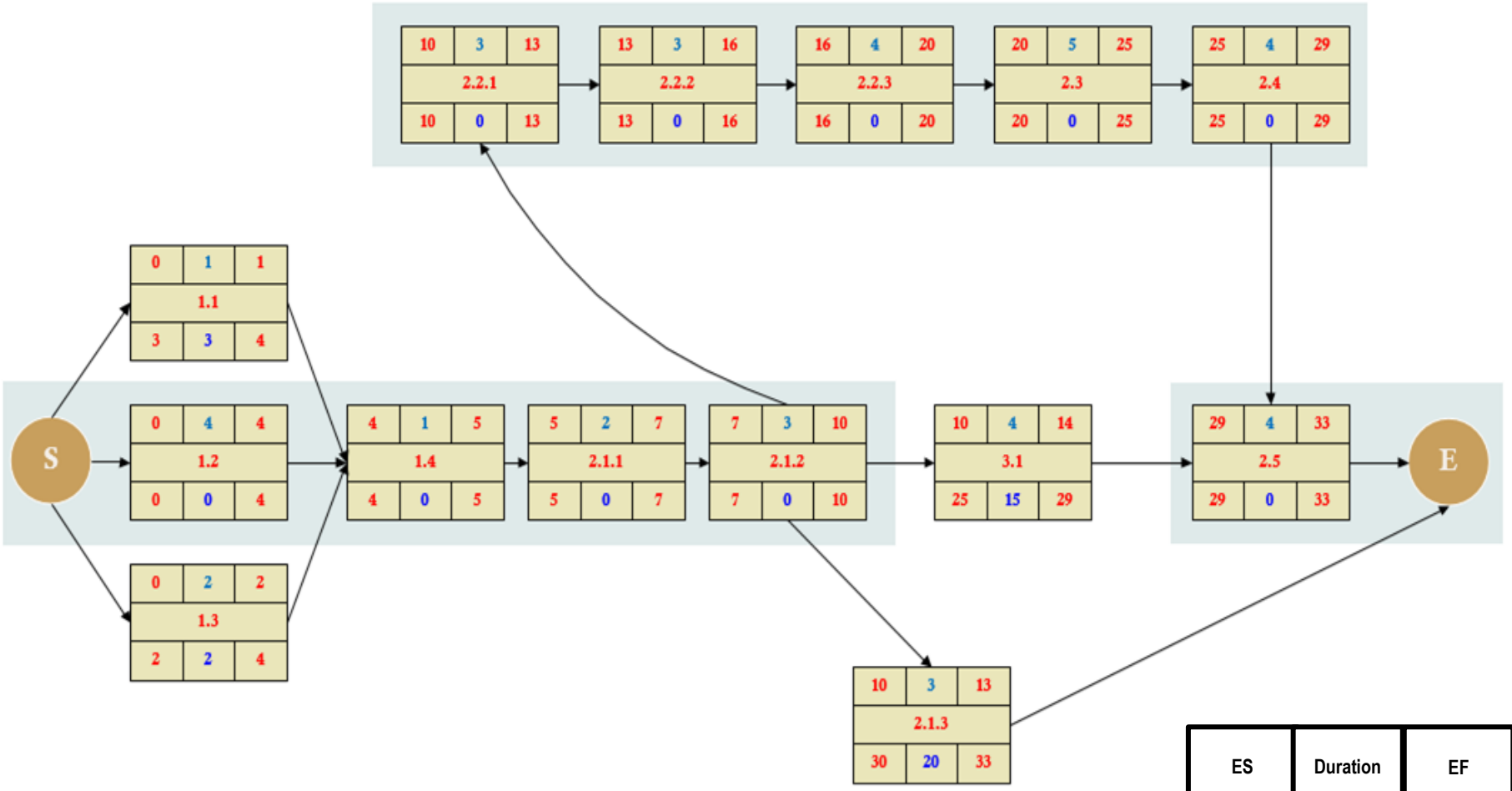
PERT Chart: activity

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



PERT Chart

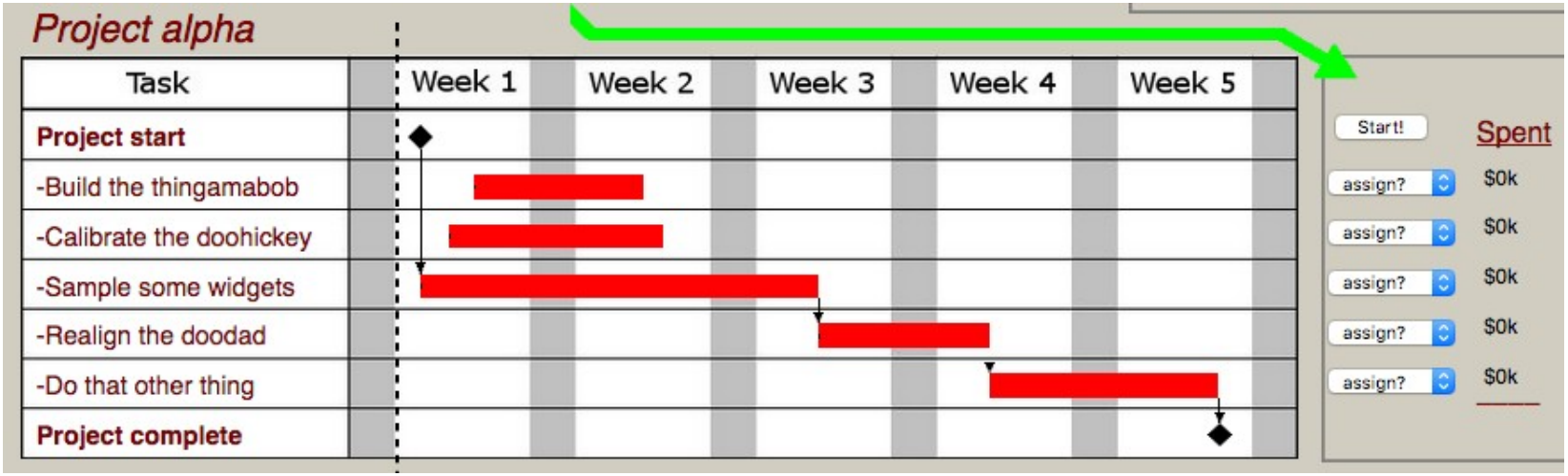


Critical Path = 1.2 + 1.4 + 2.1.1 + 2.1.2 + 2.2.1 + 2.2.2 + 2.2.3 + 2.3 + 2.4 + 2.5

Construct Resource Schedule

Play the Project Management Game:

http://thatpmgame.com/

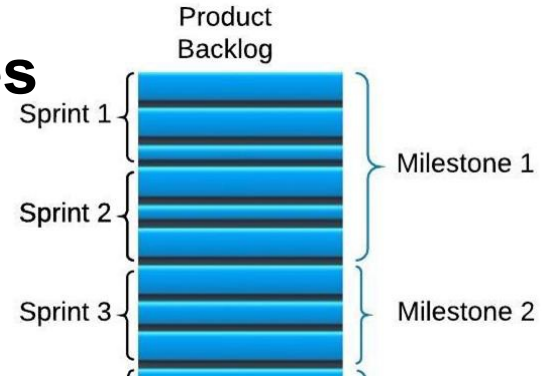


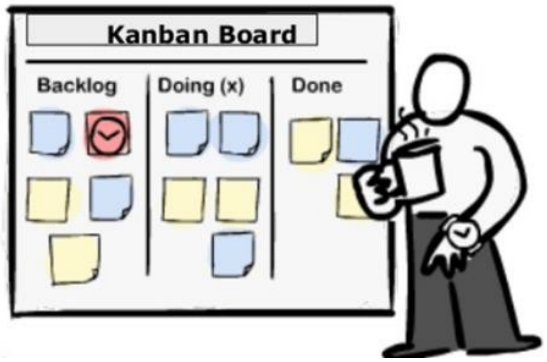
Use a Gantt chart to assign staff to various tasks.

Is the project completed on time and on budget?

Scrum Project Scheduling

1. Product Backlog with milestones





2. Sprint Backlog on Kanban board

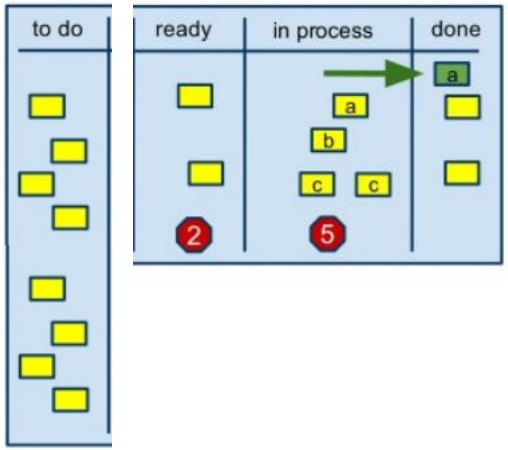
3. Burndown Charts

how to plan the schedule

Velocity and Visual Board

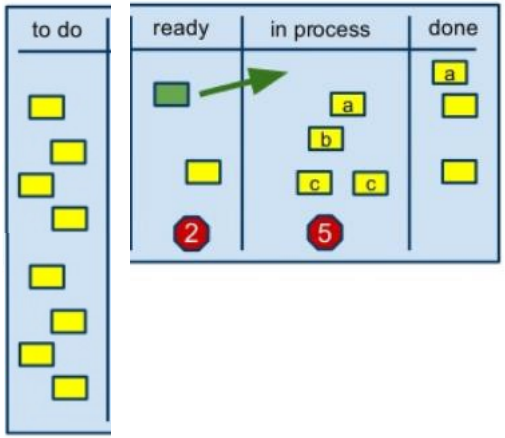
How many User Stories are ***"done"*** over the time-boxed Sprint?

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

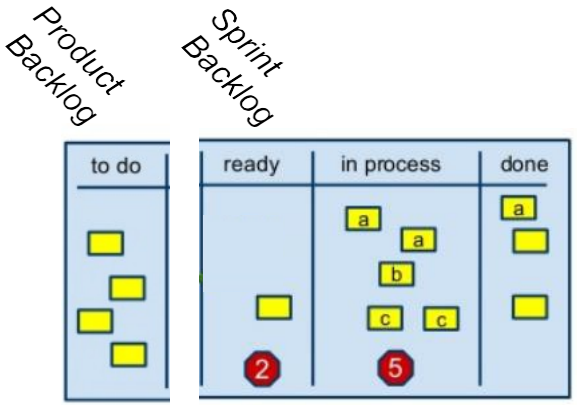


Team member A completes code for a card and moves it to "done"

Swimlane Board



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

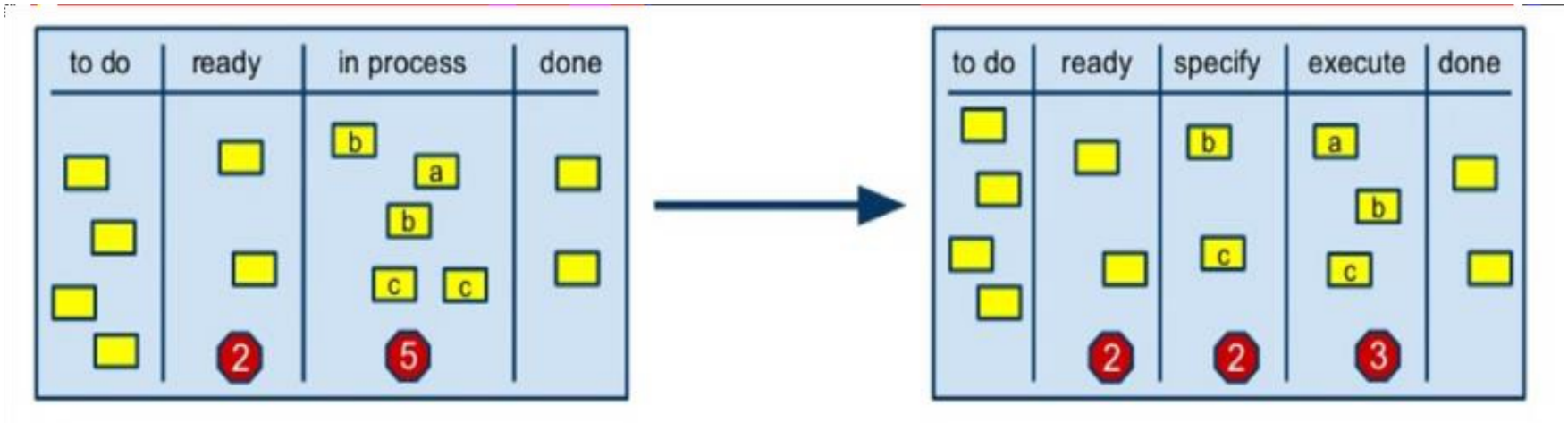


The Product Owner selects the next priority set of cards (Sprint Backlog) and moves it to "ready"

Agile Scrum Velocity

Velocity determines when dev team can deliver

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

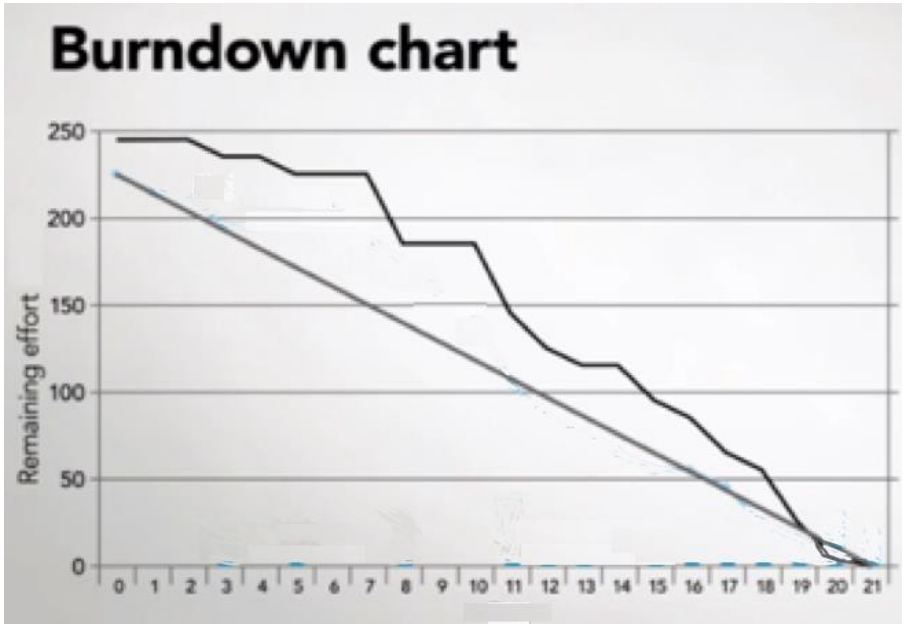


Agile Scrum Velocity

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



X-axis: time

Thank You!


HOW'S YOUR PROJECT COMING ALONG?
IT'S A STEAMING PILE OF FAILURE.
www.dilbert.com scottadams@aol.com
IT'S LIKE FIFTEEN DRUNKEN MONKEYS WITH A JIGSAW PUZZLE.
HOW'S YOUR PROJECT COMING ALONG?
FINE.
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
© Scott Adams, Inc./Dist. by UFS, Inc.

SWEN90016

Software Processes & Project Management

Teams, Virtual Teams Communication

2021 – Semester 1

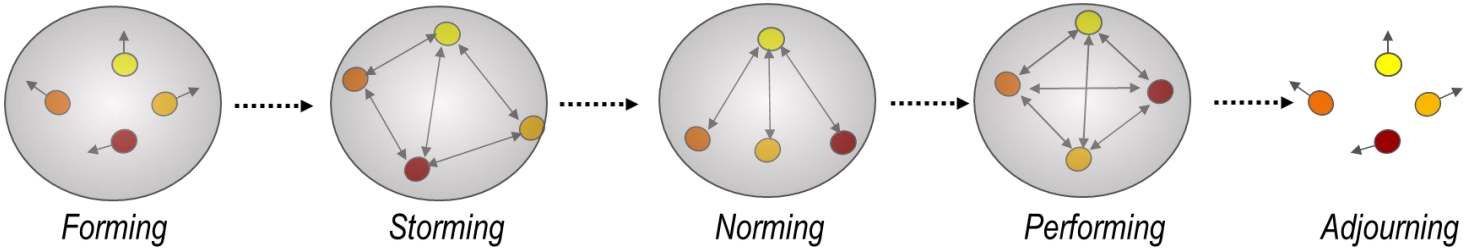
Tutorial 6

Today's aim

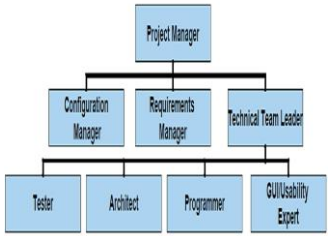
Explore **teams-** how teams are formed, and understand the different roles that individuals play in those teams.

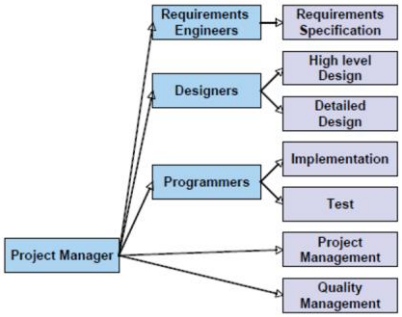
Teams Formation

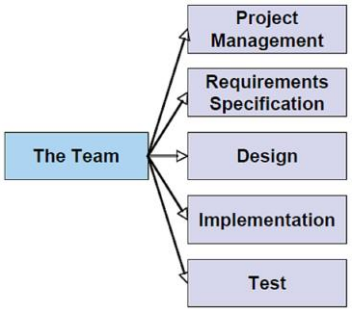
Team life cycle

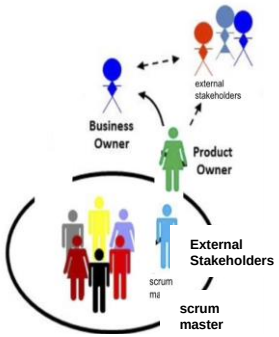


Team structures

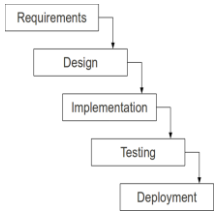


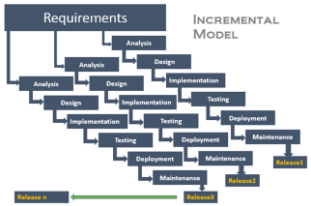


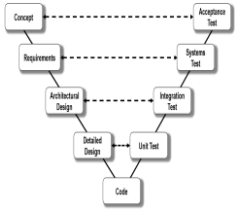


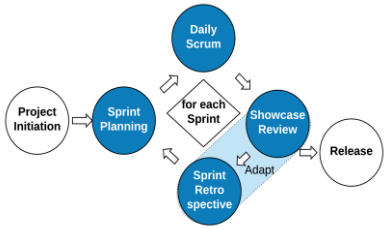


SDLC models and teams









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.

SDLC models and teams

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

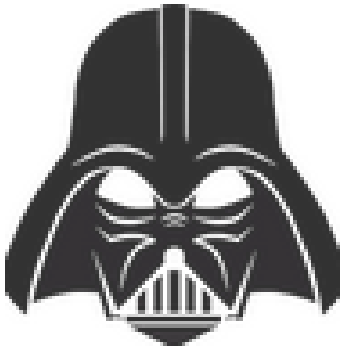
Team Roles

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



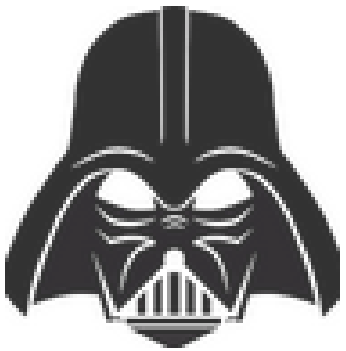
Team Roles

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



Team Roles

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

Today's aim

Explore **teams-** Communication.

Professional communication.

Virtual communication

Virtual Teams

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)

Virtual teams-communication

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

2021 – Semester 1

Tutorial 7

Today's aim

Become familiar with

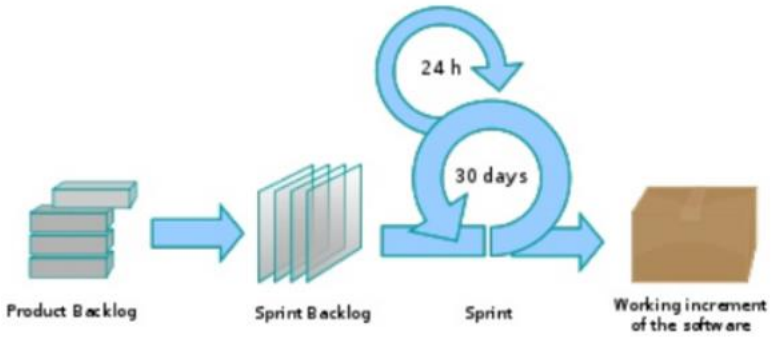
Agile

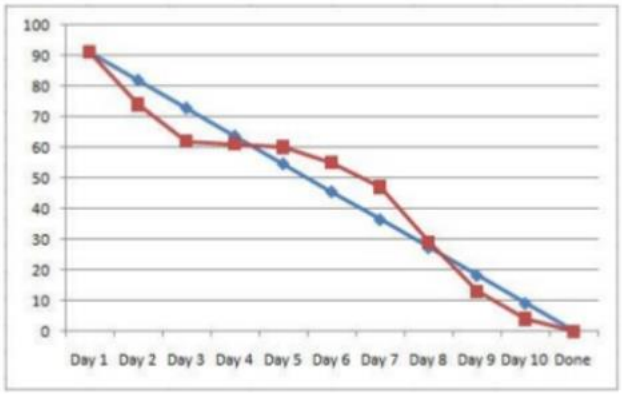
User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II

Scrum Overview





Roles

Product Owner
Scrum Master
Development Team

Ceremonies

Daily Stand Up
Sprint Planning
Sprint Review
Sprint Retrospective

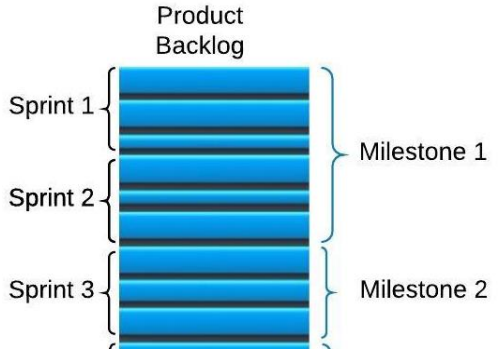
Artifacts

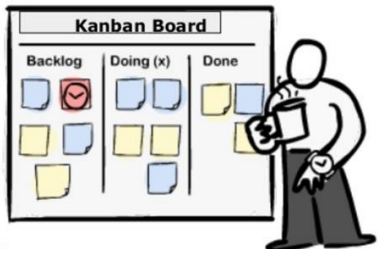
User Story
Product Backlog
Sprint Backlog
Burndown Chart
Burnup Chart

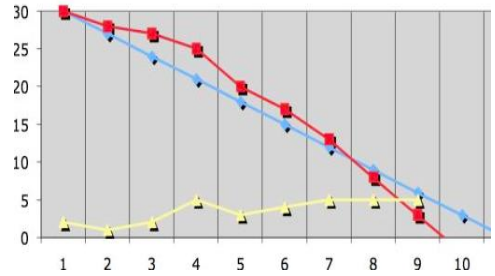
From Lecture 2, slide 51

Scrum Artifacts Overview

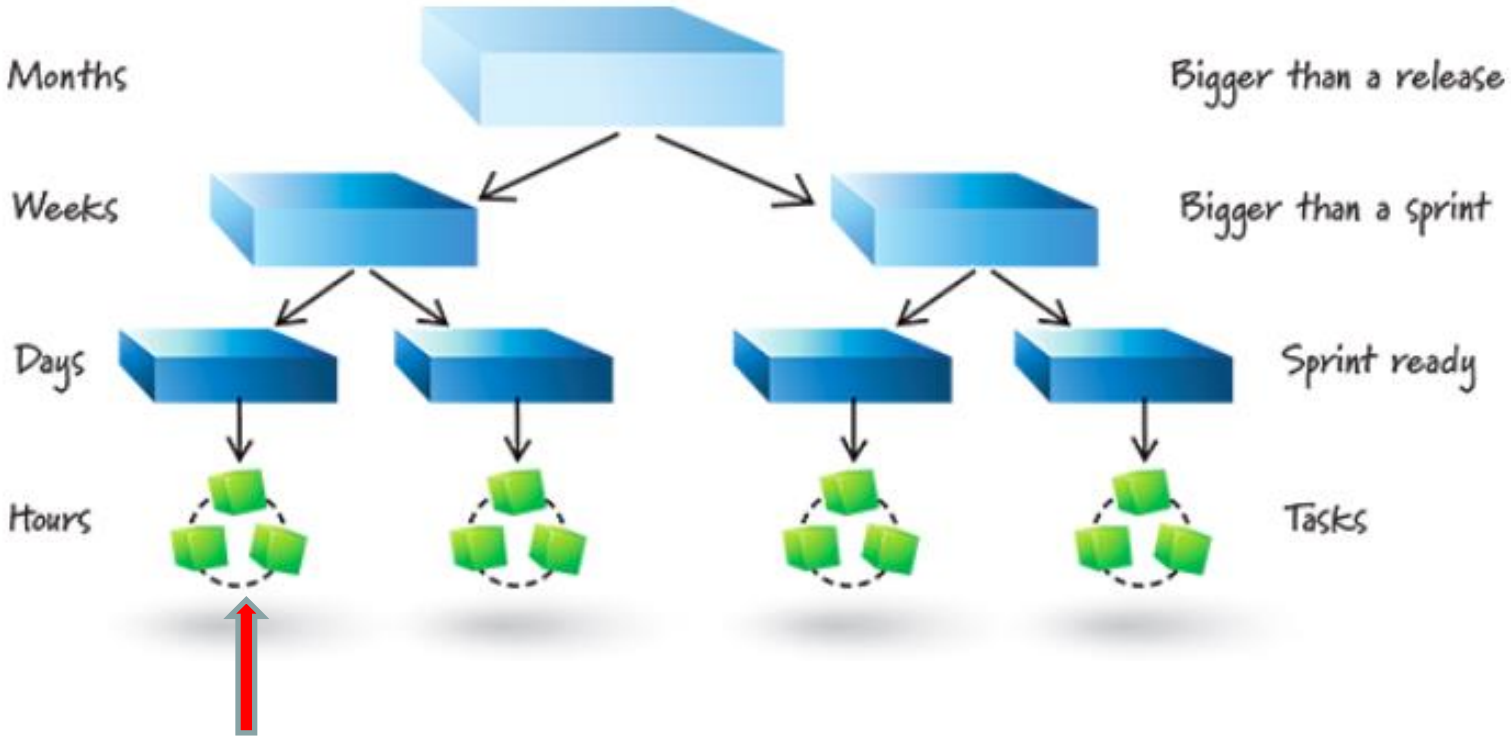
- User Stories
  - *As a <user>, I want < goal> so that < reason>.*
- Product Backlog
  - Features listed in client priority order
  - Release milestones annotated to list
- Sprint Backlog
  - Features selected for this iteration
  - Visual Kanban board
- Burn Down Chart
  - Measure the features **100%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







User Story: revision



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

User Story Size

(Sprint) **User Story**

- A developer's perspective
- A conversation placeholder

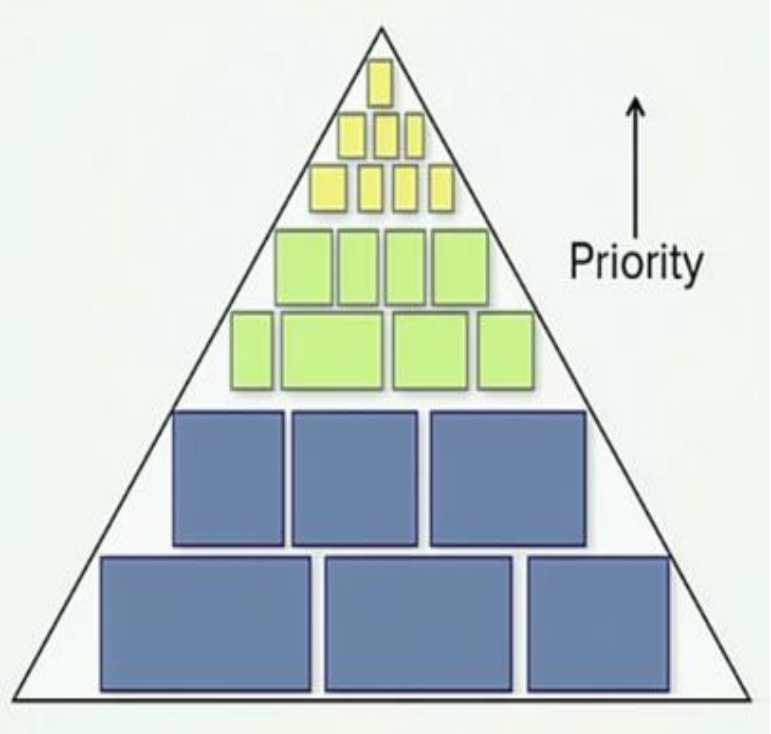
Feature User Story

- Product capabilities
- Product Owner perspective

Epic User Story

- New business services
- A product

Contentious! Advice from the internet may vary …



User Story Effort Estimation

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

User Story Effort Estimation

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.

User Story Effort Estimation

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.

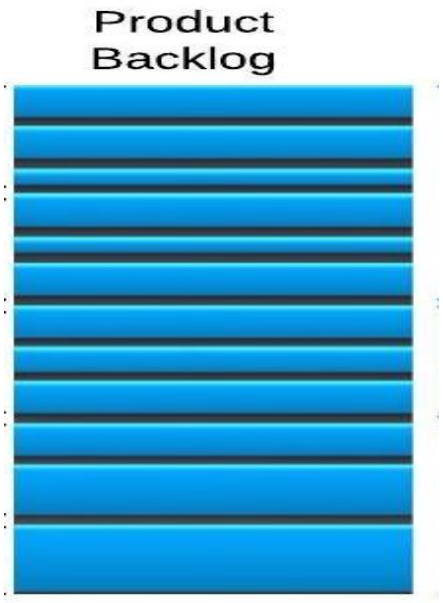
- Since I anticipate making 50 trips taking 6 minutes each, my estimate of duration is 300 minutes or 5 hours.

Scrum Release Planning

Project:
phase
Initiation

Fixed Date and Time constraints

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

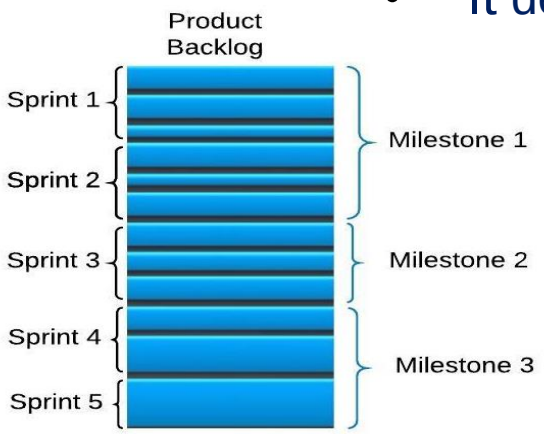


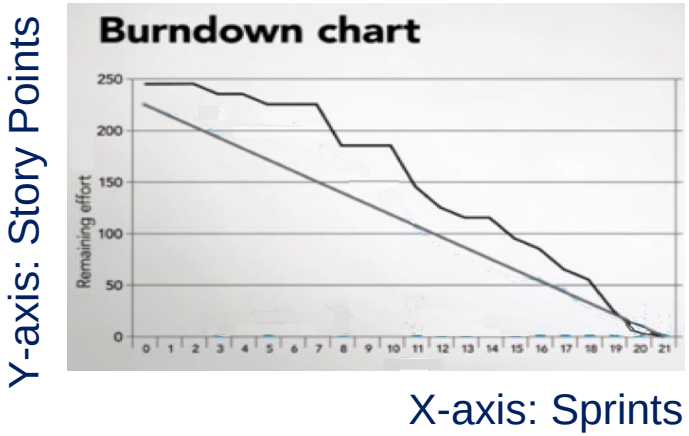
Scrum Release Planning

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





Sprint Planning

Project phase: every Sprint Planning

- Create Sprint Backlog
  - Select high value User Stories from Product Backlog
  - Use velocity to fit appropriate number of Story Points

Fixed Date and Time constraints

- 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

Sprint Planning

Fixed Scope constraints

Project phase: every Sprint Planning

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/

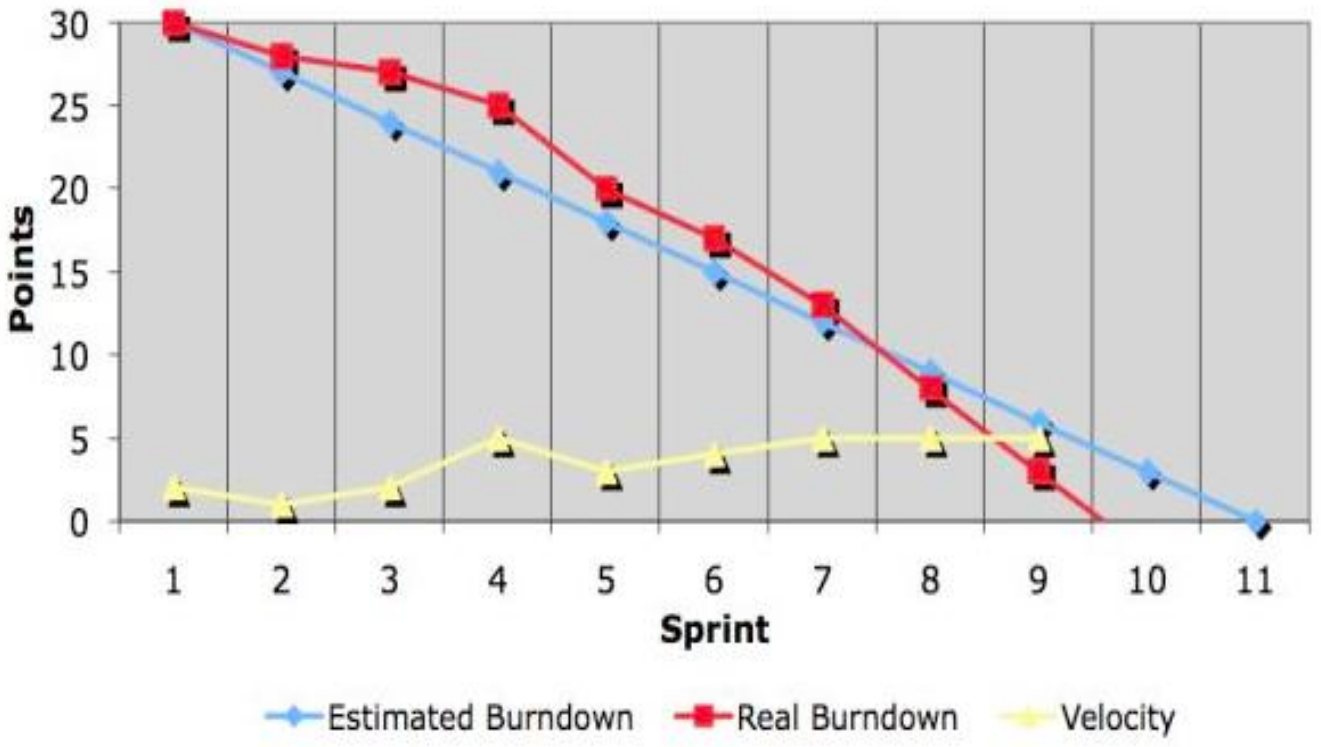
- A full task level Sprint Backlog estimated in hours is equivalent to a formal schedule (Gantt)

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

Sprint Monitoring

Fixed Date and Time constraints

Project phase: Sprint



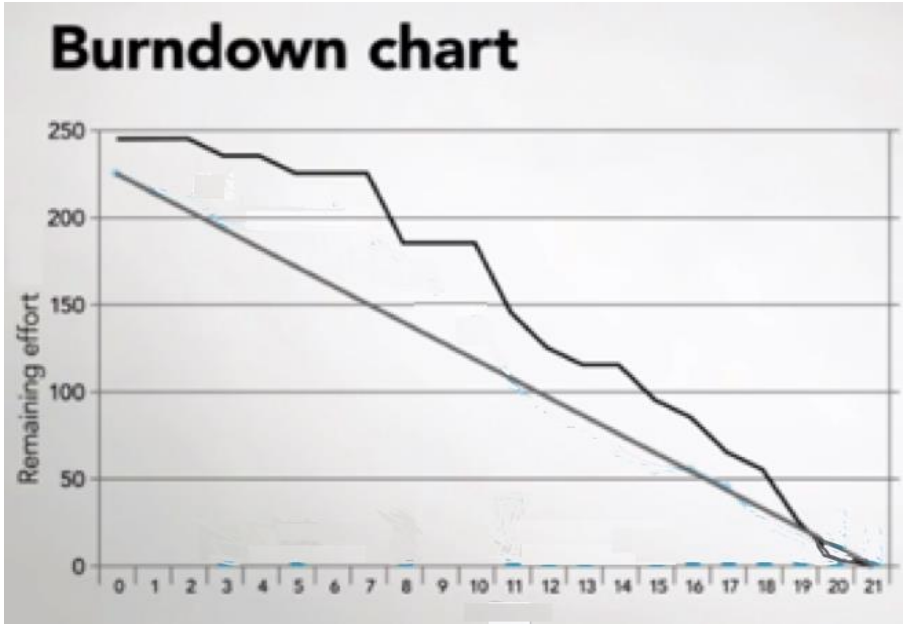
- Sprint Burn-down chart **monitors actual velocity**
- Scrum Master updated chart after daily standup

Agile Scrum Velocity

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

X-axis: time

– 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**.

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

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?

   Total = sum of SP / velocity per fortnight
   = 35 total SP / 7 velocity
   = 5 fortnights

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?

   Remaining = sum of SP / velocity per fortnight
   = (1+ 27) / 8
   = 3.5 fortnights
   = 7 weeks
   New Total = 7 + 2
   = 9 weeks

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)

Estimation Strategy Overview

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

Today's aim

Become familiar with

Agile

User Stories and Story Points and Velocity

Formal

Function Point Analysis and COCOMO II

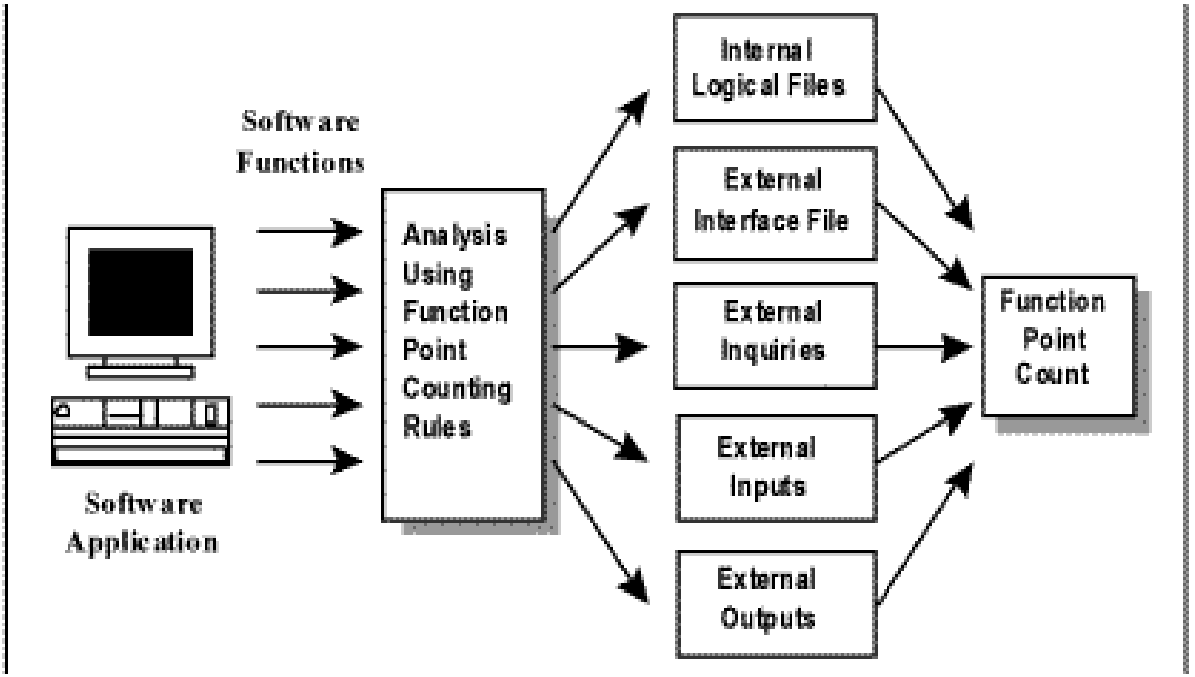
Functional Points

What are they?

PMBOK

Historic Data

Done at any time in project lifecycle



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

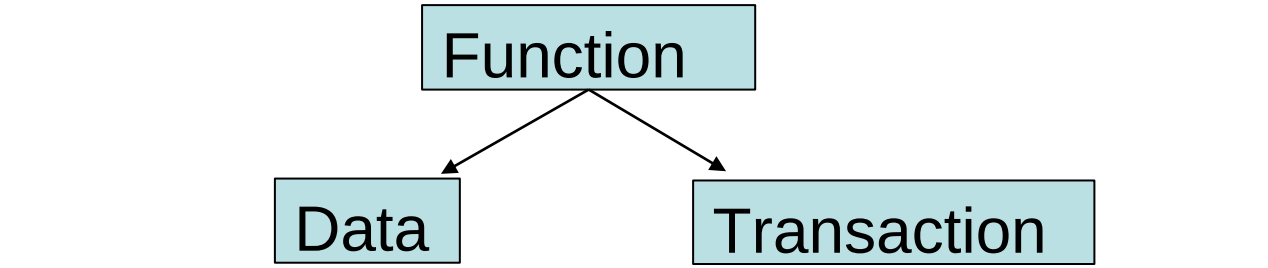
1. Categorize functional requirements and count

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)



Data elements
Record elements

Data elements
File references

Step 2: Set Complexity Values

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

Step 3: Calculate Functional Points

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

Step 4: Calculate VAF

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

Step 5: Calculate Adjusted FP

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 Adjusted Functional Points

COCOMO II – another strategy

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

Quality Assurance
Planning, Control, Monitoring

2021 – Semester 1

Tutorial 8

Today's aim

1. Understand testing in Agile
2. Testing in Formal - checklists for your team

Agile Quality Strategy

What is the role of QA in agile?

a) After development there is a separate testing done by the agile team in a number of sprints

b) As there is fast development cycles there is no time for testing

c) Agile aims to adapt to changes quickly and minimize time so there is no testing

d) Testing is done in each sprint

e) Continuous integration between development and testing

Testing in Agile

Every sprint has its own testing phase.

The tests can be ran every time new features are released.

In Agile testing, small piece of working software are delivered to the customer at the end of the sprint.

Testers and developers work closely together in Agile testing. Testing is done by the whole team.

User acceptance is performed at the end of every sprint.

The User story describes the requirement …
The acceptance criteria provides the definition of when that user story is done.

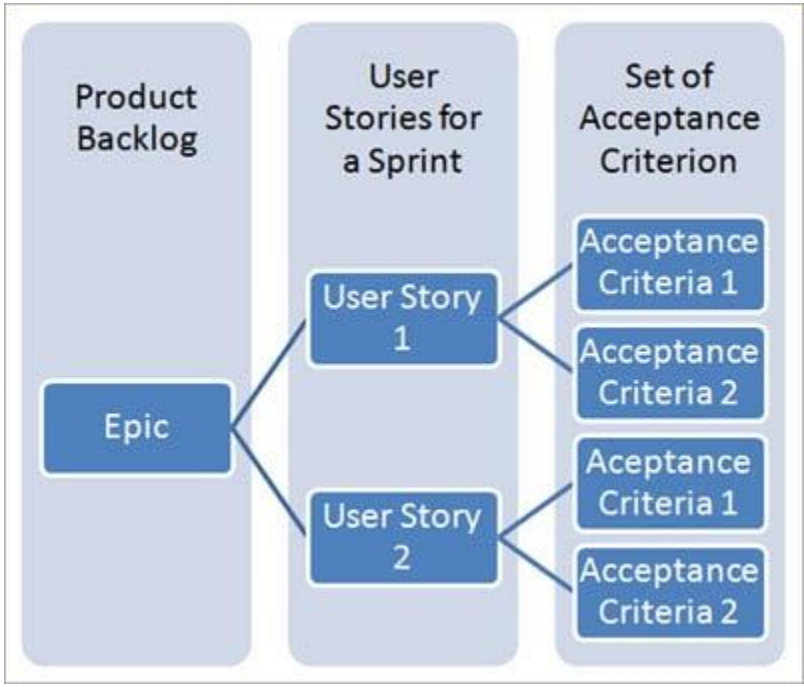
Acceptance Criteria

As a Customer… I want to be able to split my payments…
So I can pay using multiple debit cards

Activity: In your breakout groups, brainstorm three (3) Acceptance Criteria for the above User story

Examples:

- ***User can select 'split payment' on the payment page***
- ***User can choose different types of payment options***
- ***User can specify the amount they want to split***
- ***The payments page automatically calculates what is left to pay as user enters their split payment***



https://www.softwaretestinghelp.com/user-story-acceptance-criteria/

Acceptance Criteria

Examples:

- ***As a User I can select 'split payment' on the payment page so that I can choose to use multiple cards when I want to pay***
- ***As a User I can choose different types of payment options so that I can pay***
- ***As a User I can specify the amount I want to split between payment options so that I can use multiple cards when I want to pay.***
- ***As a User the payments page should automatically calculate what is left to pay so that the user knows how much to is left to pay when the user enters their split payment***

https://www.softwaretestinghelp.com/user-story-acceptance-criteria/

Agile Quality – Dev Team process

Agile QA desk-audit hurdle:

Invite multi skilled audience to desk audit: a business analyst, another developer and a tester

Review the code at the developer's desk, before the code is allowed to be committed into the shared *git* repository, GITHub.

Once the code is committed into GITHub, it's test suite is run immediately by the Continuous Integration tool

CI tool displays run code's pass/fail status

Agile Quality Process

Sprint Review QA evaluation:

What
Where
Who
When
Why

- 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 ***technical debt item*** in the Product Backlog, (optionally in FDD format)
- The Product Owner sets the priority of the ***technical debt item***

QA Requirements

Write QA Requirements as User Stories

As the Agile Scrum team:

5 a. *We want a Quality Plan, so that our Sprint has a strong Quality Management focus*

As a Quality Assurance **Design** team:

b. *We want a QA checklist, so that key categories and attributes are assessed at defined times*

As the System Administrator:

c. *I want a password policy guideline, so that our application has helpful processes*

I want a password policy checklist, so that our application is highly secure

Acceptance Criteria

Another example

User story: *As a user, I want to be able to recover the password to my account, so that I will be able to access my account in case I forgot the password.*

Scenario: Forgot password
Given: The user has navigated to the login page
When: The user selected *forgot password* option
And: Entered a valid email to receive a link for password recovery
Then: The system sent the link to the entered email
Given: The user received the link via the email
When: The user navigated through the link received in the email
Then: The system enables the user to set a new password

https://www.softwaretestinghelp.com/user-story-acceptance-criteria/

Quality Plan: A checklist

A tool

Quality Attributes	Definition According to McCall et al.
Correctness	The extent to which a program satisfies its specifications and fulfils the user's mission objectives.
Reliability	The extent to which a program can be expected to perform its intended function with required precision.
Efficiency	The amount of computing resources and code required by a program to perform a given function.
Integrity	The extent to which access to software or data by unauthorised persons can be controlled.
Usability	The effort required to learn, operate, prepare input, and interpret output of a program.
Maintainability	The effort required to locate and fix an error in an operational program.
Testability	The effort required to test a program to ensure that it performs its intended function.
Flexibility	The effort required to modify an operational program.
Portability	The effort required to transfer a program from hardware and/or software environment to another.
Reusability	The extent to which a program (or parts thereof) can be reused in other applications.
Interoperability	The effort required to couple one system with another.

Another checklist

A tool

Checklist for software requirements specification artifact

Organisation and Completeness

- ☐ Are all internal cross-references to other requirements correct?
- ☐ Are all requirements written at a consistent and appropriate level of detail?
- ☐ Do the requirements provide an adequate basis for design?
- ☐ Is the implementation priority of each requirement included?
- ☐ Are all external hardware, software, and communication interfaces defined?
- ☐ Have algorithms intrinsic to the functional requirements been defined?
- ☐ Does the specification include all of the known customer or system needs?
- ☐ Is the expected behaviour documented for all anticipated error conditions?

Correctness

- ☐ Do any requirements conflict with or duplicate other requirements?
- ☐ Is each requirement written in clear, concise, unambiguous language?
- ☐ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ☐ Is each requirement in scope for the project?
- ☐ Is each requirement free from content and grammatical errors?
- ☐ Is any necessary information missing from a requirement? If so, is it identified as "to be decided"?
- ☐ Can all of the requirements be implemented within known constraints?
- ☐ Are any specified error messages unique and meaningful?

Quality Attributes

- ☐ Are all performance objectives properly specified?
- ☐ Are all security and safety considerations properly specified?
- ☐ Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

Traceability

- ☐ Is each requirement uniquely and correctly identified?
- ☐ Is each software functional requirement traceable to a higher-level requirement (e.g., system require-

Verification Process

In your breakout groups, **discuss** quality processes & do these activities

2. Create an appropriate formal checklist to review the group assignment

3. Describe the outcome of this review?

Done!

SWEN90016

Software Processes & Project Management

Ethics, Outsourcing, and Procurement

2021 – Semester 1

Tutorial 9

Ethical Responsibilities of IT

	Australian Computer Society (ACS) Code of Professional Conduct	IEEE: Software Engineering Code of Ethics, Professional Practice
1	Priorities: place the interests of the community above personal or sectional interests. Preserve the integrity and security of the other's information.	Public: Software engineers shall act consistently with the public interest.
2	Competency: work competently and diligently for my clients and employers. Advise when I believe a proposed project is not in their best interests	Client and Employer: act in the best interests of their client & employer, consistent with the public interest.
		Product: Software engineers shall ensure that their products meet the highest professional standards possible.
3	Honesty: be honest about my skills, knowledge, services and products. Not knowingly mislead a client as to the suitability of a product or service	Judgment: Software engineers shall maintain integrity and independence in their professional judgment.
4	Social Implications: I must strive to enhance the quality of life of those affected by my work. Respect people's privacy.	Management: promote an ethical approach to the management of software development.
		Profession: advance the integrity and reputation of the profession, consistent with the public interest.
		Colleagues: be fair to and supportive of their colleagues.
5	Professional Development: enhance the professional development of myself, colleagues, employees, students and be aware of community issues affecting the IT profession.	Self: participate in lifelong professional learning and promote an ethical practice of the profession.
6	Information Technology Profession: enhance the integrity of the IT profession and respect each other. Take appropriate action if I discover a colleague has unethical behavior.	

Ethical Responsibilities of IT

Class Activity

In your breakout groups:

- Examine the ACS Code of Professional Conduct and compare with IEEE Software Engineering Code of Ethics.
- How are the two codes similar/different?

Ethics: Practical Guide

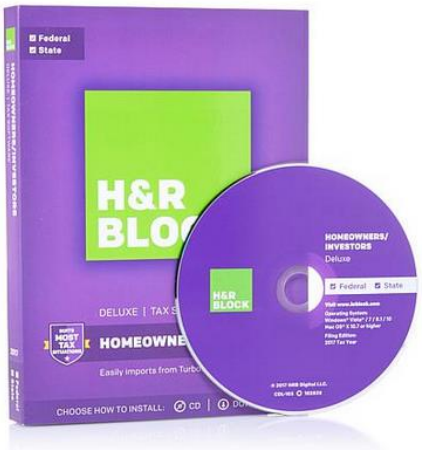
Questions to ask & consider before making a decision:

1. Would I be happy for this action to be prominent in tomorrow's news?
2. Is there a universal rule that applies here?
3. Will the proposed action result is a good outcome?
4. What would happen if everybody did this?
5. How will this action impact on the character of myself/ my organisation?
6. Is the action consistent with my values and principles?

Lecture 5, Slide 10

Tax Pack Case Study

Ethics Case Study 1 – Tax Software Package



In your breakout groups, evaluate the IT ethical dilemma.

- As the president, what would you have done?
- How could the ACS code of ethics have guided you?
- What is the relationship between the ethical and the legal?

Outsourcing

The practice of engaging an external party (under contract) to perform services or create goods that are traditionally performed in house by the company's own employees.

Types of Outsourcing:

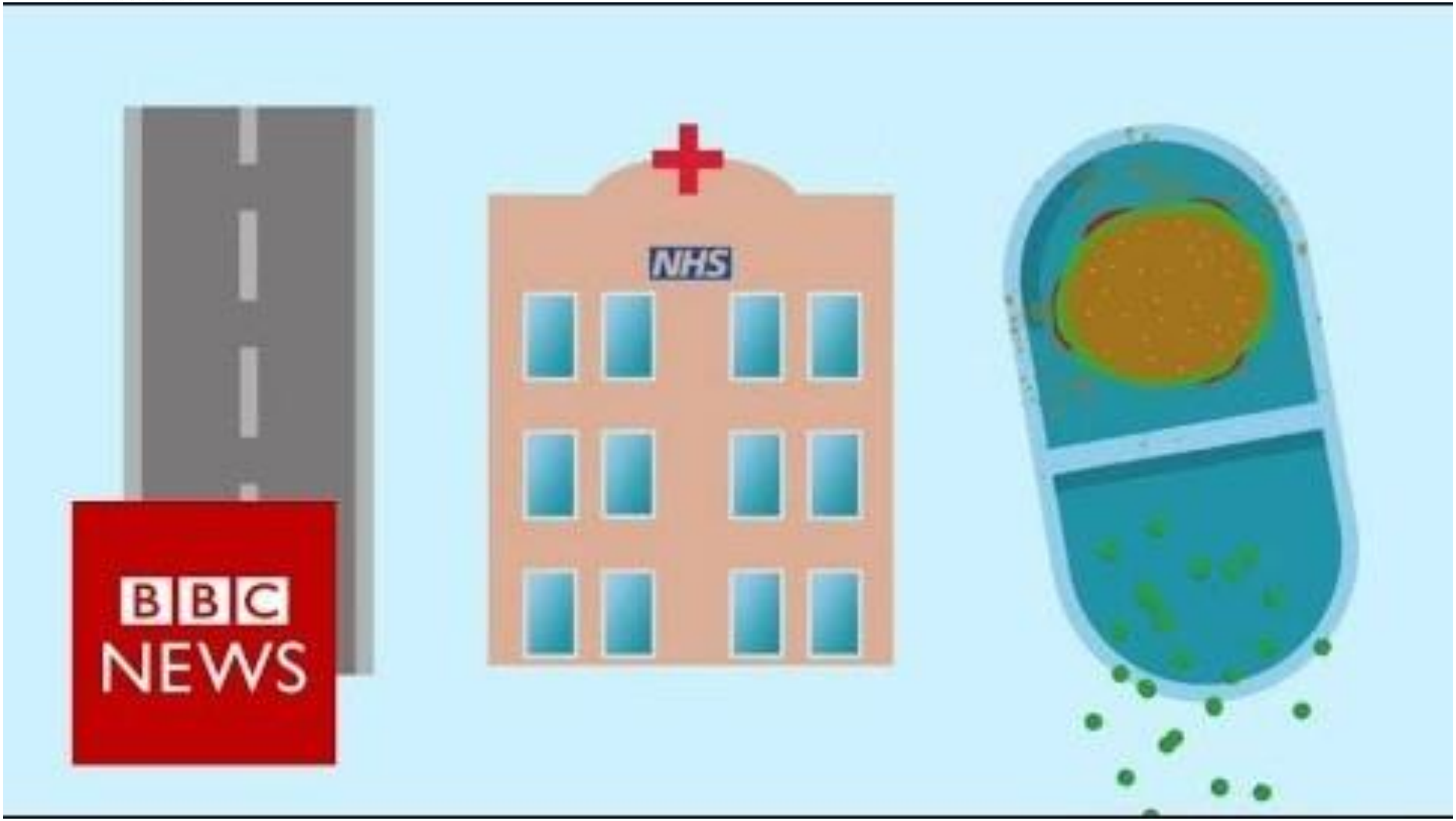
1. Onshoring:
•Relocating activities inside national borders to access targeted benefits.

2. Nearshoring:
•Activities relocated to another country with close proximity e.g. New Zealand, Indonesia.

3. Offshoring:
•Activities relocated to another country irrelevant of geographical location and time zones.

Lecture 9, Slide 18

Outsourcing



https://www.youtube.com/watch?v=TTAr_J53x70

Outsourcing: Pros and Cons

Pros

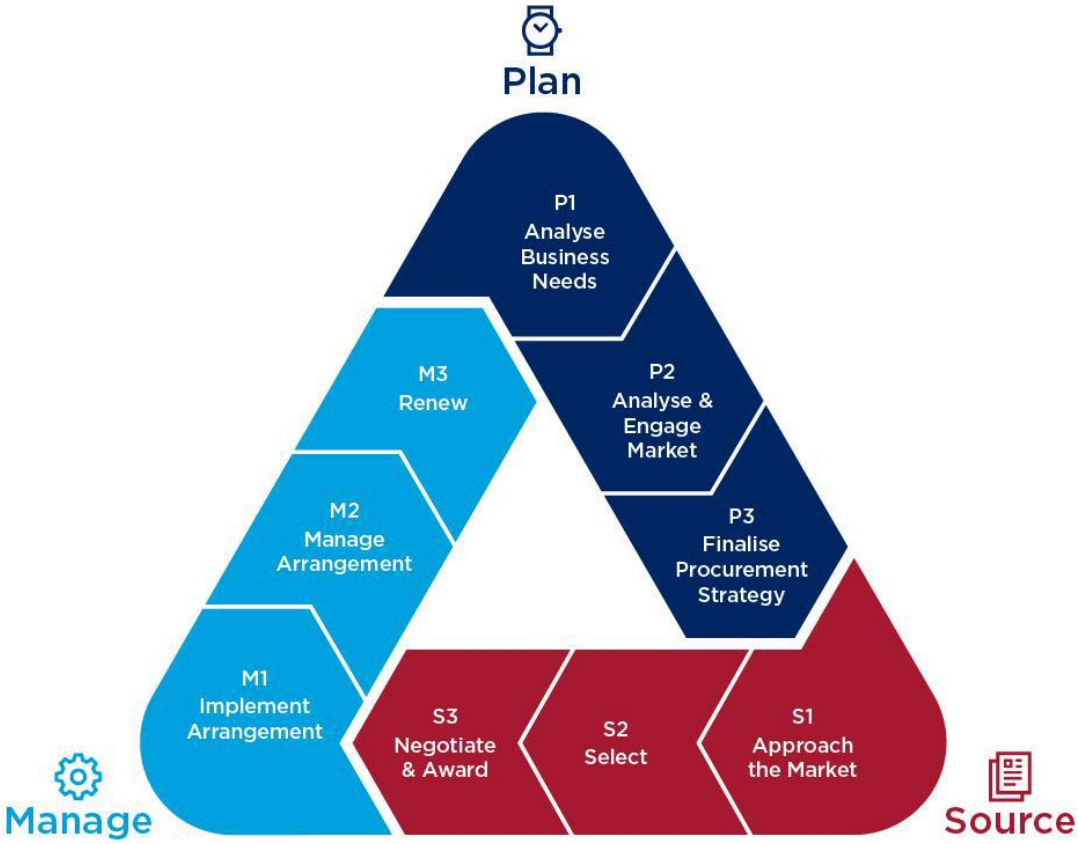
- Reduces costs
- Access to difficult to find capabilities & skills
- Time savings –24/7 based activities
- Freeing scares internal resources to focus on core business activities
- Leverage best practice
- Access to better Technology
- Lower training costs in high turn over jobs
- Flexibility –Ramp up and down
- Increased Accountability -Contracts
- Risk mitigation –Access established and proven approaches e.g. Agile, Project Management etc

Cons

- Loss of control
- Process / supply chain fragmentation
- Security issues
- Employees feel threatened
- Additional effort and cost to engage and manage
- Lower quality work / work to contract
- Time zone, cultural & language challenges
- Location stability -Political, Economic, Religious
- Ethical standards -environment, slave / child labour
- Difficult to change
- Damages to the local job markets
- Loss of Relationship building opportunity with key stakeholders

Lecture 9, Slide 20

Procurement



- Traditional waterfall

- Agile

www.procurepoint.nsw.gov.au/policy-and-reform/nsw-government-procurement-information/nsw-procurements-approach

Contracts: Procurements

Formal PMBOK: Plan and document fixed scope

Find answers to high level 5W's

What
Where
Who
When
Why

- The buyer prepares a detailed Statement of Work (SoW)
- The buyer prepares a Request for Proposal (RFP) or Quote (RFQ)
- The seller/buyer sign a contract, include the SoW
- Contract types vary: "fixed price" (seller risk), "time & materials" (buyer risk)
- The quality metrics are based on a Service Level Agreement (SLA) contract

Thank You!

SWEN90016

Software Processes & Project Management

Configuration and Version Control

2021 – Semester 1

Tutorial 10

Today's aim

Become familiar with

Continuous Integration

&

GIT

https://github.com/microsoft

Continuous Integration (CI)

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.

So what is Git?

- 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

So what is Git?

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!

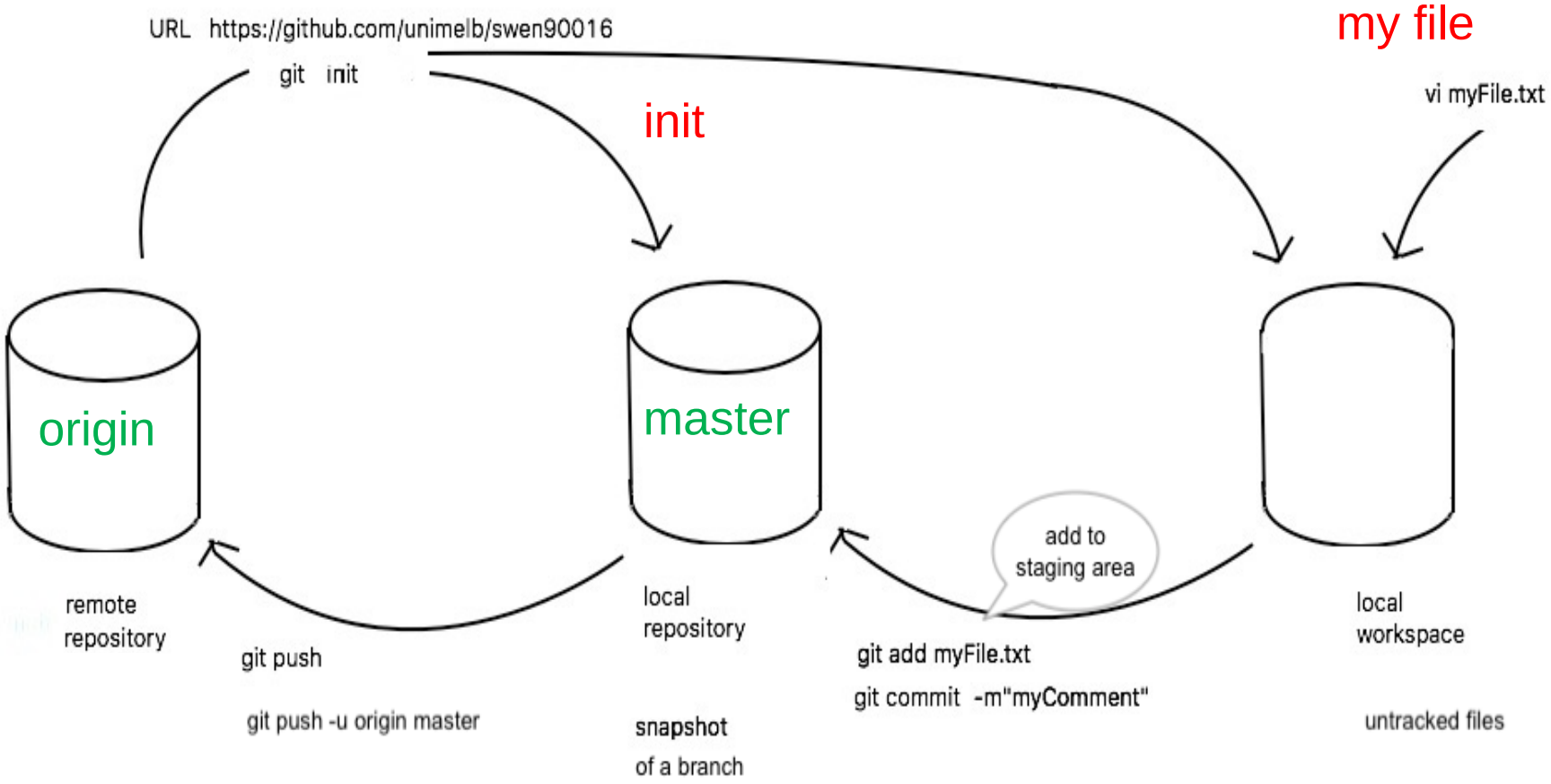
Git Advantages

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

General Git Workflow

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**

git Round Trip



github

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.

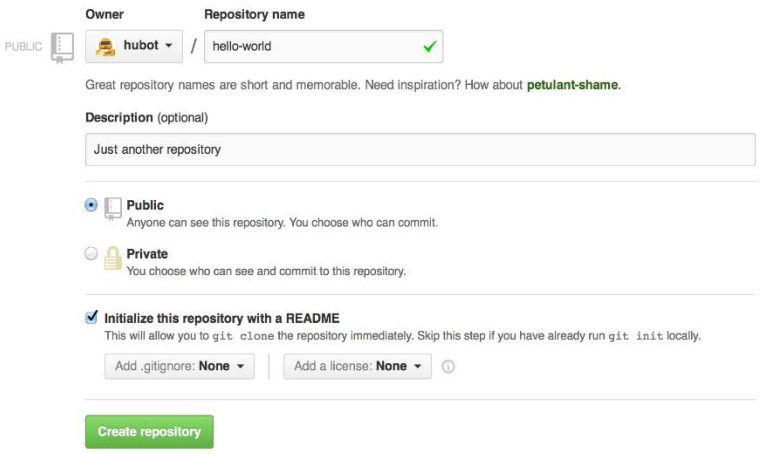
To Create a New Repository

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**.



github- branching

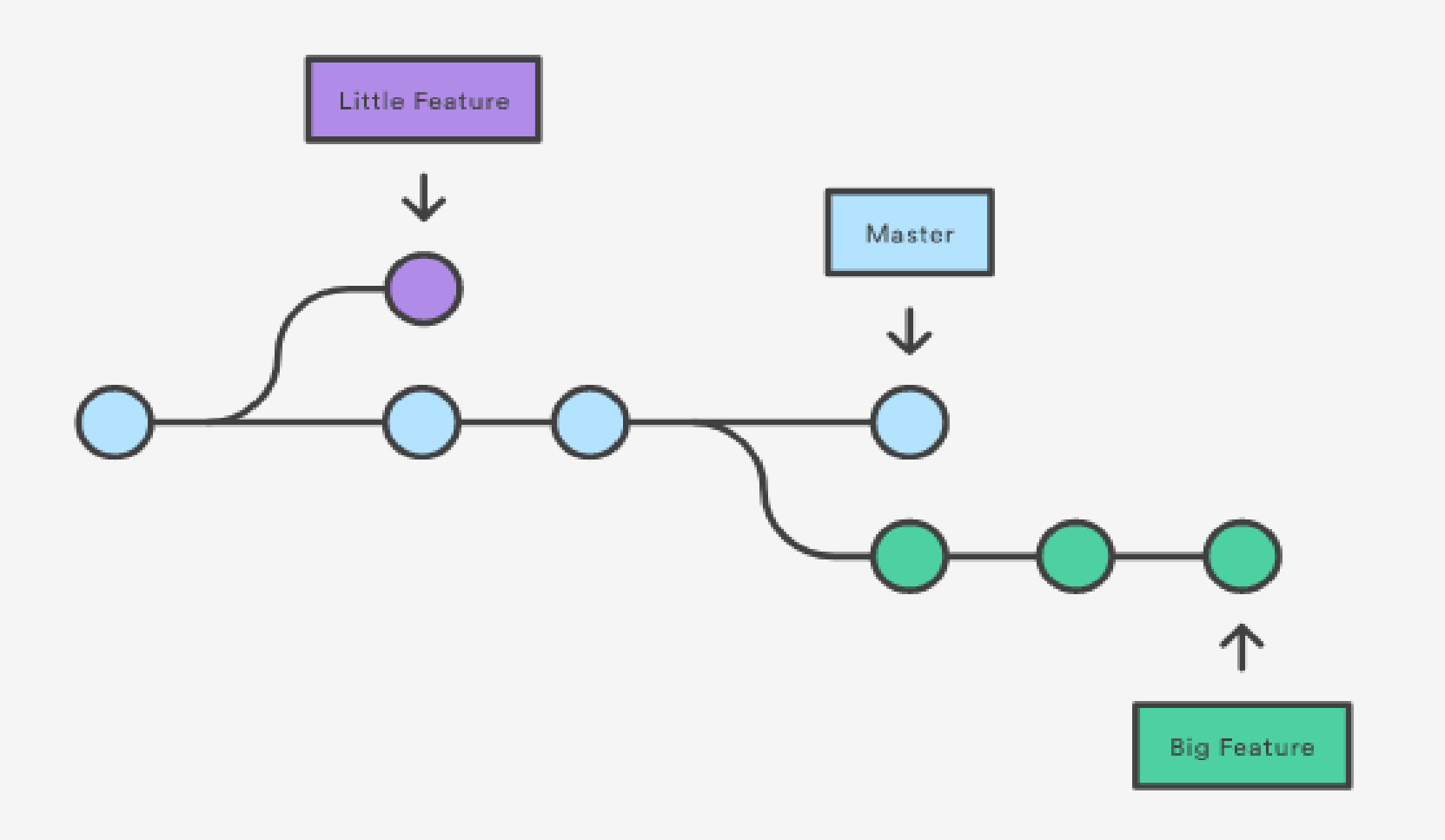
Step 2. Create a Branch

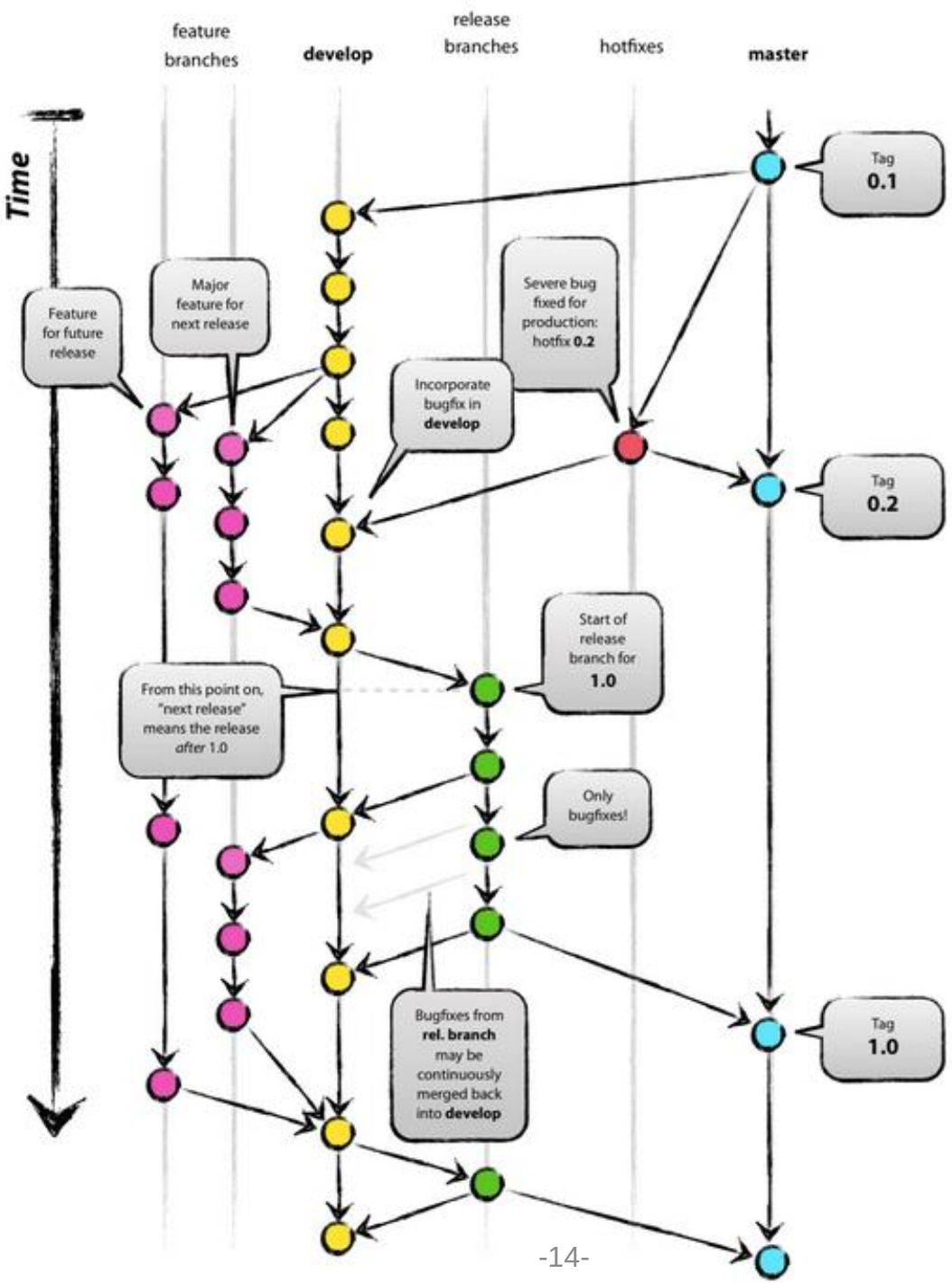
Branching is the way to work on different versions of a repository at one time.

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.

git concept: **branching**




feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

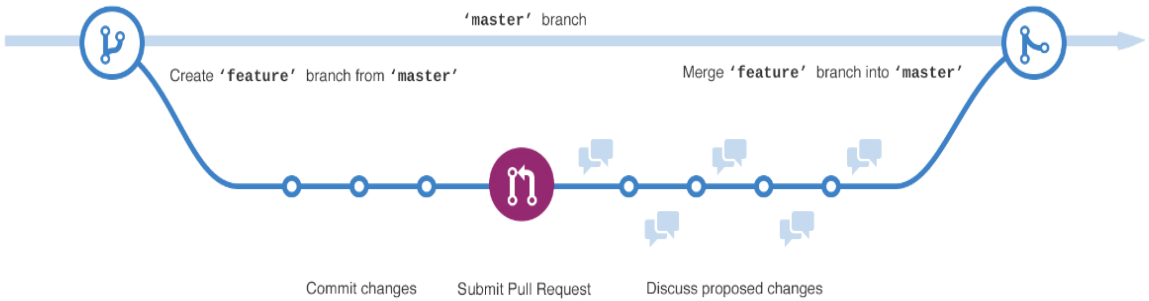
Step 2. Create a Branch

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:

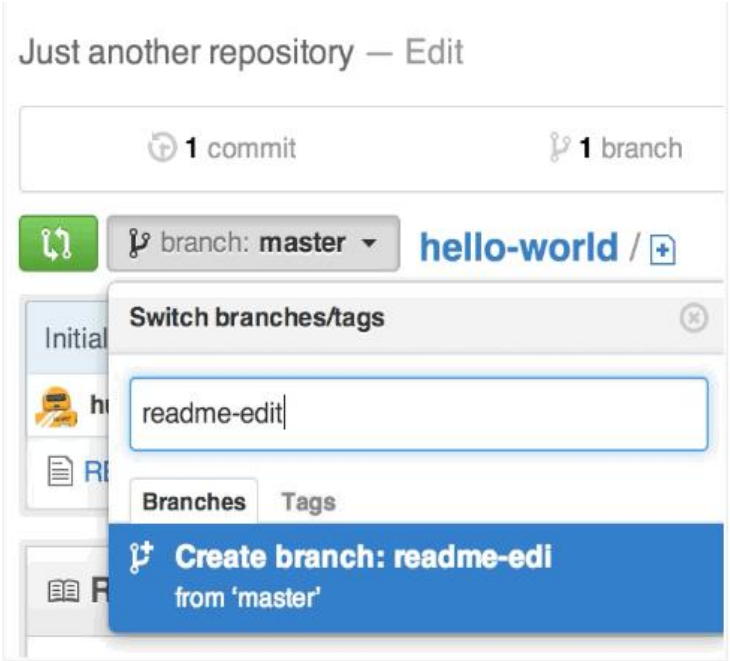
story.txt
story-joe-edit.txt



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

github



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.

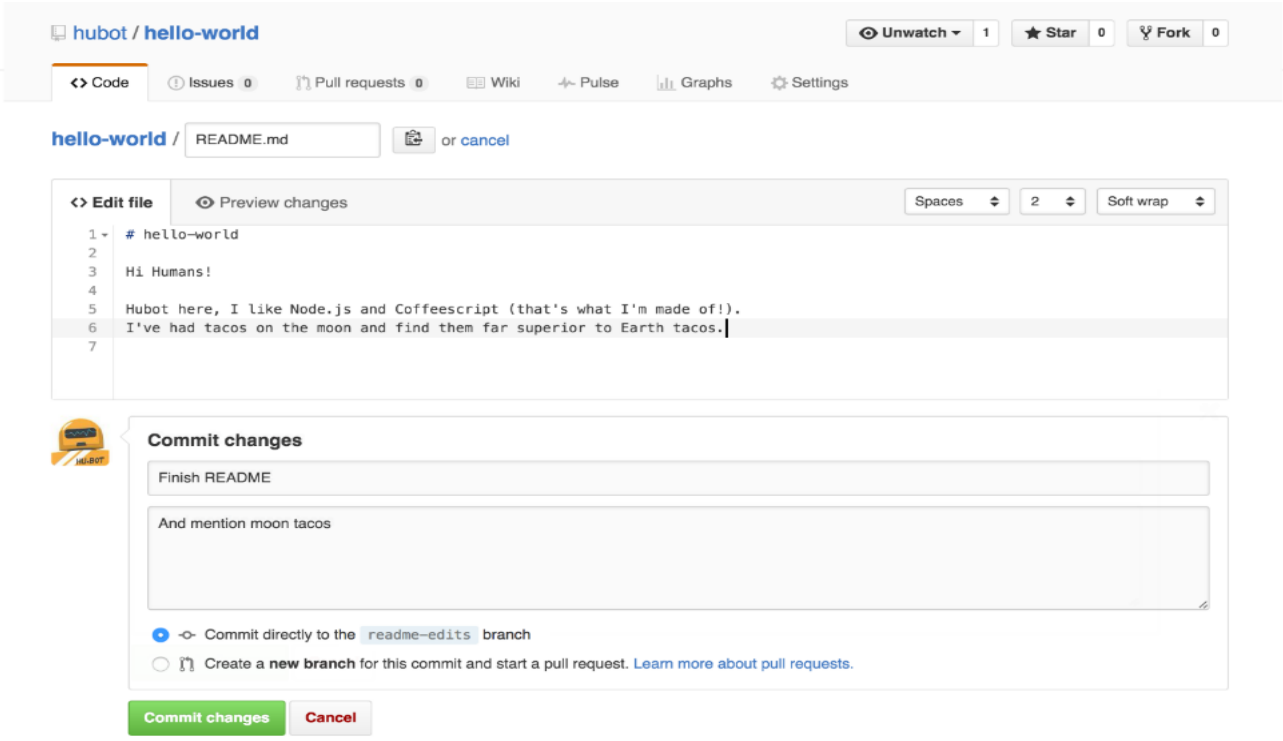
In the editor, write a bit about yourself.

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*.

github


hubot / hello-world
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Wiki
Pulse
Graphs
Settings
hello-world / README.md
or cancel
Edit file
Preview changes
Spaces
2
Soft wrap
hello-world
Hi Humans!
Hubot here, I like Node.js and Coffeescript (that's what I'm made of!).
I've had tacos on the moon and find them far superior to Earth tacos.
Commit changes
Finish README
And mention moon tacos
Commit directly to the readme-edits branch
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit changes
Cancel

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.

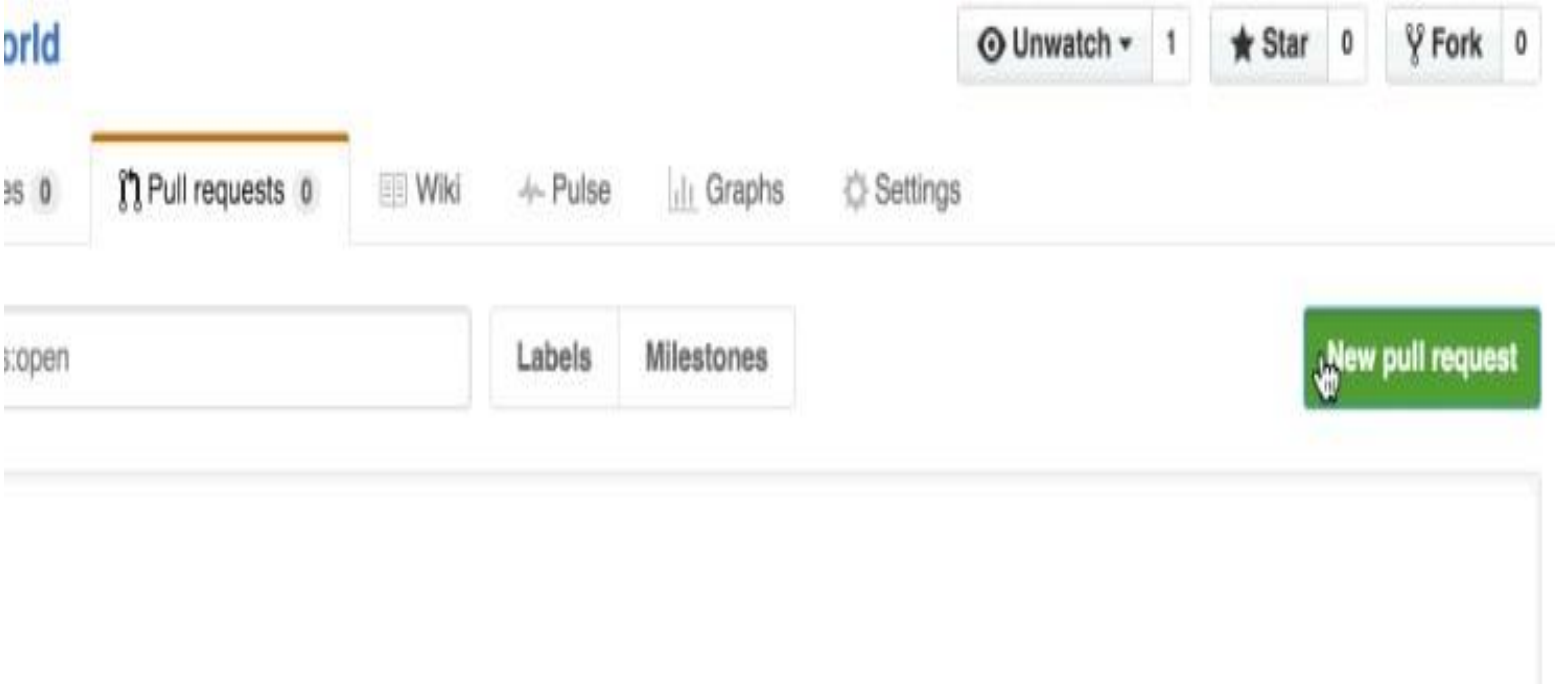
As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

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.

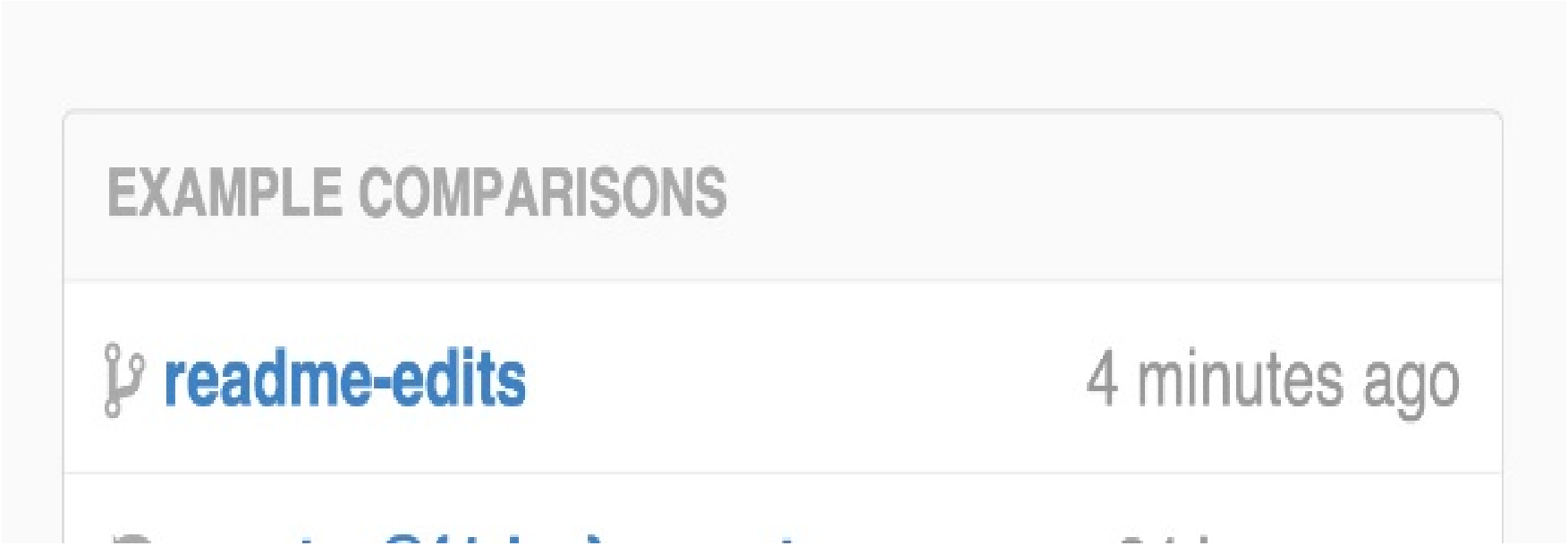
Open a Pull Request for Changes to the README

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



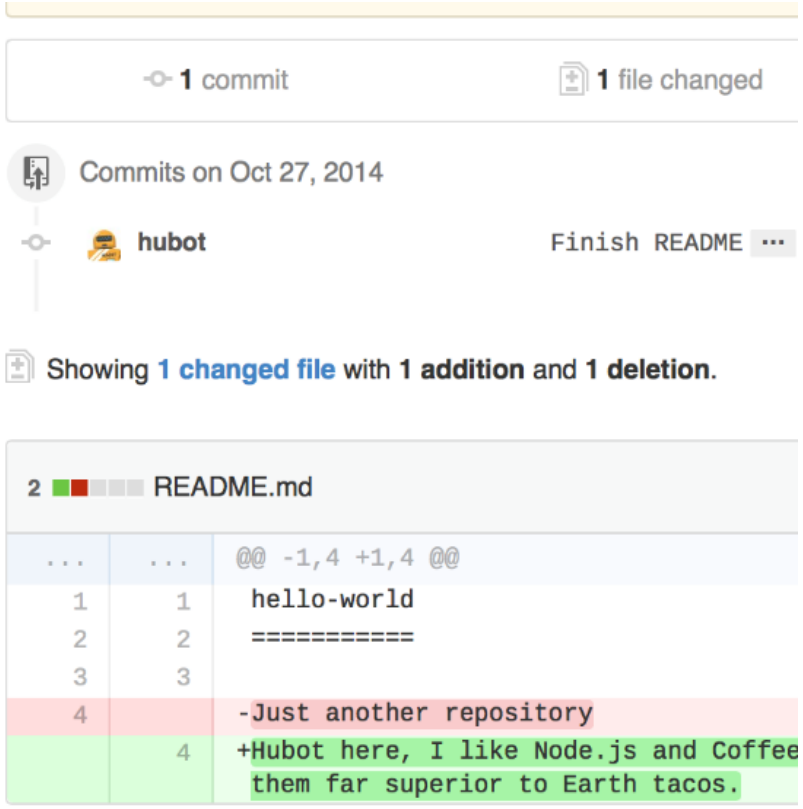
Open a Pull Request for Changes to the README

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



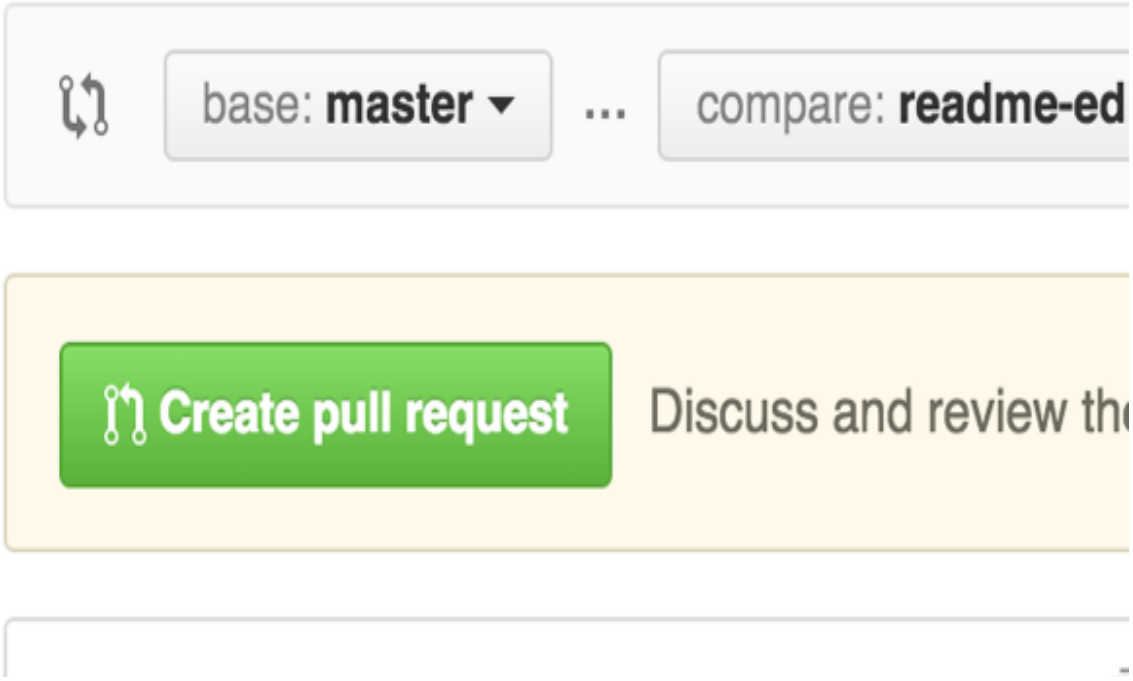
Open a Pull Request for Changes to the README

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



Open a Pull Request for Changes to the README

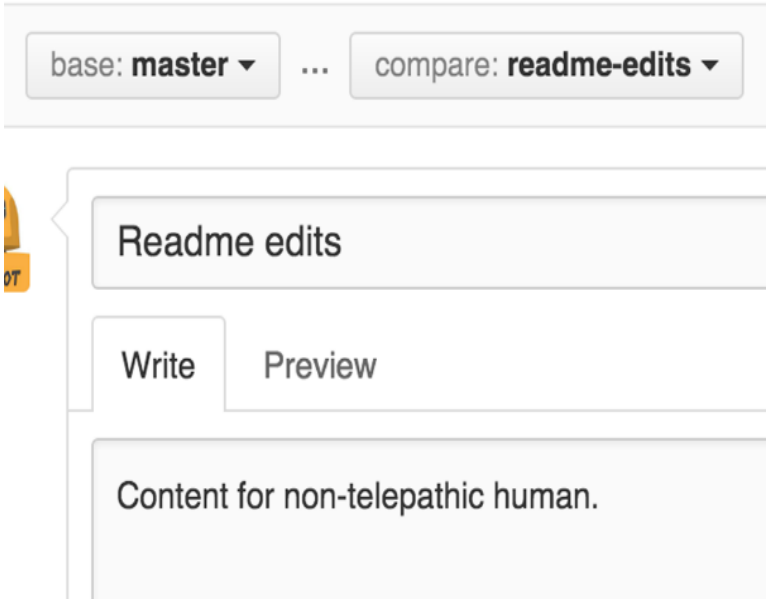
When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.



Open a Pull Request for Changes to the README

Give your pull request a title and write a brief description of your changes

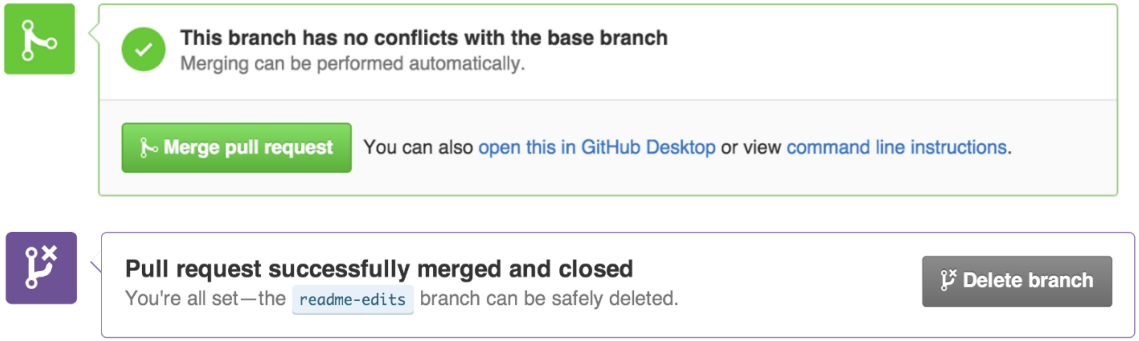
When you're done with your message, click **Create pull request**!



Step 5. Merge Your Pull Request

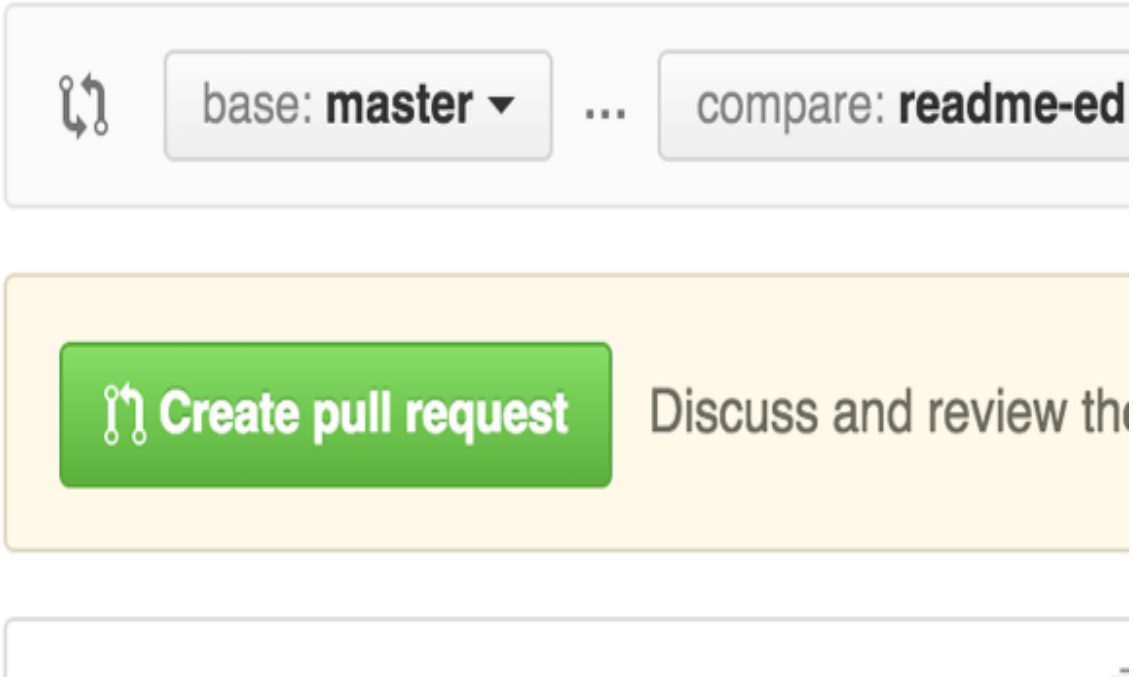
In this final step, it's time to bring your changes together – merging your readme-edits branch into the master branch.

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.



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.



Assignment 2- Group Presentations

Week 12- check the shared doc for your timeslot to present

What to expect:

- All team members need to be present
- Nominate a team member to share their screen and present the website
- You have 5-7 minutes to present
- You will be tested on 4 main areas:
  1. Create a customer (0.5)
  2. Make a voucher booking- check email (0.5)
  3. Login as admin (0.25)
  4. Cancel an voucher booking- (0.5)
  5. Login as admin – check it has been cancelled (0.25)

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://git-scm.com/

http://thehackernews.com/2015/03/github-hit-by-massive-ddos-attack from_27.html