**Module 2**

**Projects:**

Definition: 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 (Male/Female, cost, time)

Why to 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

**Project Management:**

Definition: 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: (advantages)

• 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

**Project Manager Skills / Attributes:**

Definition: 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

JOB AD example (P11)

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

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

Leading

• Setting team direction

• Owning & coordinating activities across different organisational functions

• Motivating team members

• Assigning work

Controlling

• Defining project baselines

• Tracking project progress

• Project status reporting

• Determining and taking corrective actions

Agile Scrum Master Key Activities

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

**Module 3**

**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

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

PRICNE 2

• 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

How to select one:

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

**Project Success / Failure**

• 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.

Success determining factors (P19)

**Project Screening and start** (P21-22)

**Business Case**

Purpose: 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

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

Role and Responsibility (P27)

**Investment Techniques**

Include:

• Return On Investment

• Net Present Value

• Payback period

• Rough Order of Magnitude

Return On Investment (ROI)

ROI is income divided by investment

• ROI = (total discounted benefits – total discounted costs) / 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%)

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

Payback period

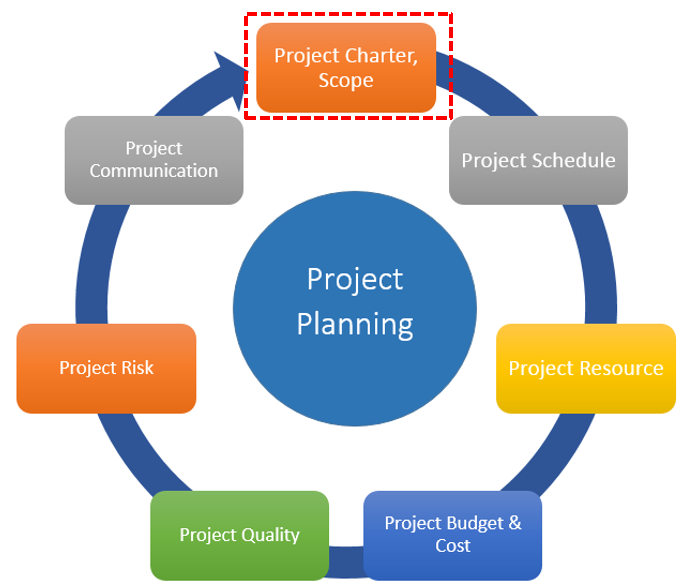
• The payback period is the amount of time it takes a project before the accrued benefits surpass accrued costs or how much time an investment takes to recover its initial cost

• 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

Estimation Rough Order of Magnitude (ROM) (P33)

**Project Charter**



**T1:**

Initialization Phase

1. Business needs analysis (analyze case study) (goal, Key Characteristics, challenge, risk)

2. Analyse constraints (scope time cost)

3. Stakeholder analysis (develop project charter)

**Module 4：**

**Project Management Plans （PMP）**

Process: A process is a series of progressive and interdependent steps by which an end is attained.

Relationship between process and 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.

Project Charter Vs 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).

Project Management Plan (Formal):

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.

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

• Project Governance

– Roles and Responsibilities

– Mandatory Project Planning / Key Additional Activities

o Schedule

o Risk Management

o Cost Estimation

o Quality Assurance

o 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.

**Module 5：**

**SDLCs:**

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:

o Delivery and Release - Deployment

o Maintenance

1.Formal Processes

• Waterfall

• Incremental

• V-Model

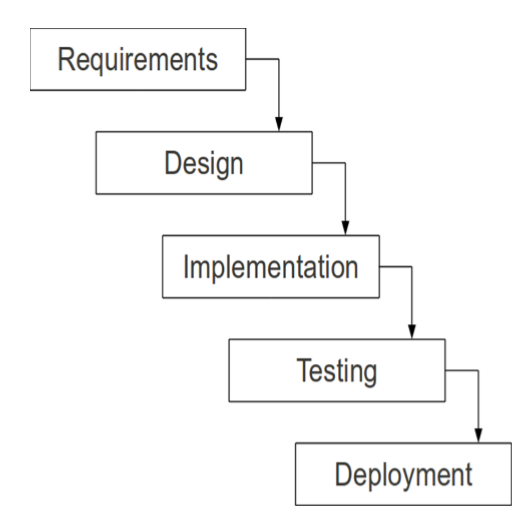
2. Agile Processes

• Scrum

• Kanban

• Extreme Programming

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 in underway

• One phase must be completed before moving on to the next

• Unclear requirements lead to confusion

• Clients approval is in the final stage

• Difficult to integrate risk management due to uncertainty

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.

Advantages

• 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

• 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

Formal Models Characteristics:

• 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 wellknown

• Large scale applications and systems developments

**Module 6:**

Agile Characteristics:

• 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!

What is agile

• A set of methodologies 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

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

Agile Framework

Primary elements

• Manifesto • 12 Key Principles • Kanban • Scrum

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.

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.

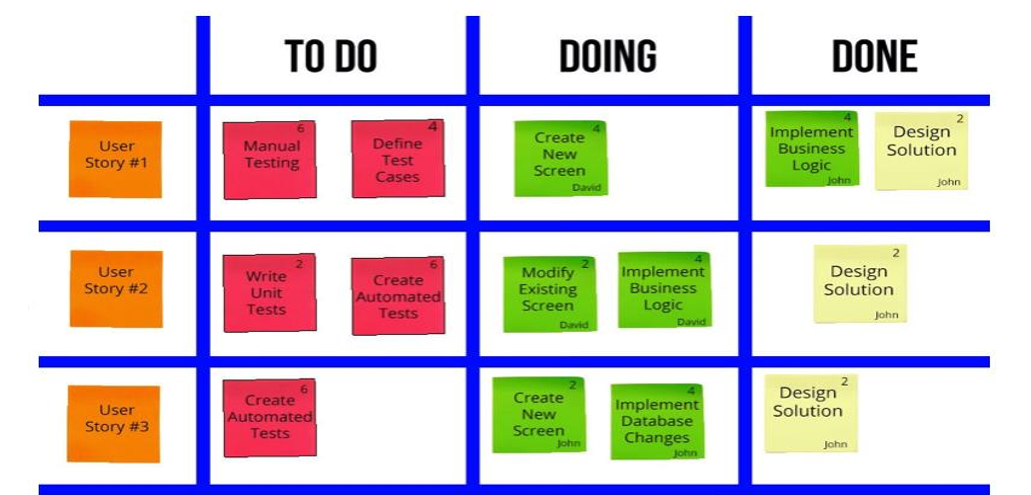
11. The best architectures, requirements, and designs emerge from self-organising teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

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

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



Scrum

Scrum is an agile way to manage a project

Definition:

•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.

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

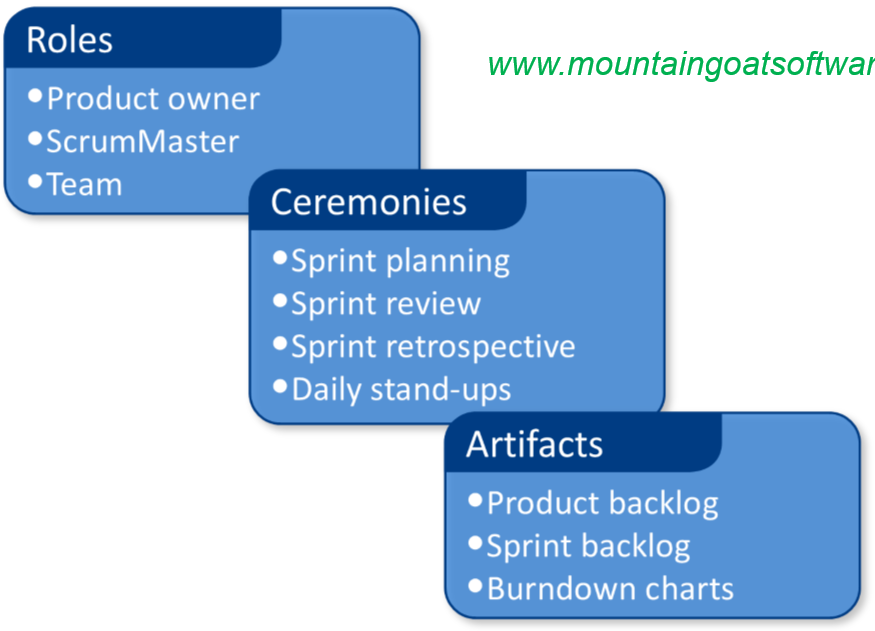
• Time frame is contained to a manageable size (weeks or months)

Sprints:

Rather than doing one thing at a time...

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

Scrum Framework:



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

– 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

– 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)

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

– 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

• Parameters

o Daily

o 15-minutes and no more than 30 mins

o Stand-up

• Not for problem solving / Not a status meeting

o Whole world is invited

o 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.

- 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

-Sprint Retrospective

• Periodically look at what is and isn’t working

• Typically 30 minutes

• Done after every sprint

• Whole team participates:

o ScrumMaster and Team

• Possibly Product Owner, customers and others

• Discuss what to:

o Start Doing, Stop Doing and Continue Doing

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 >

• 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 can get all information associated with my professional membership

• 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

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

– Product Backlog

• 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

• 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

– 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.

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)

Formal vs Agile

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?

**T2:**

Waterfall:

Advantages:

• Simple and easy management

• Rigid and sequential

• Documentation produced

• Requirements stable and precise

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

Advantages:

• Smaller, easier modules

• Initial modules released earlier

• Client feedback known earlier

• Change has less impact

• Requirements stable and precise

Disadvantages:

• Management complexity

• Increased cost

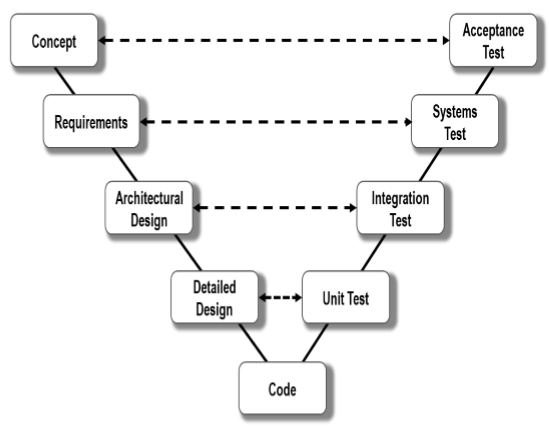
• Partition skill

• Integration risk

• Rigid within each partition

V-Model:

waterfall model plus defined artifact deliverable development stage 🡨-> testing phase



Advantages

• Simple and easy management

• Rigid and sequential deliverable

• Documentation produced

• Requirements stable and precise

Disadvantages

• Requires discipline

• Bad news known late in process

• Client feedback known late in process

• Discourages change

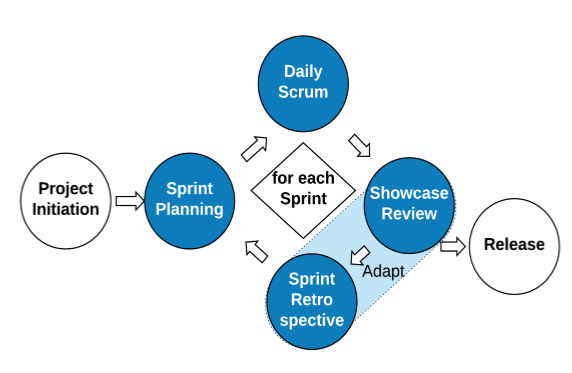
• Test artifacts are expensive

• Risks and changes have big impact

Agile:

Scrum: method to organize working teams

XP: method to improve code quality



Advantages

• Transparent productivity due to fast releases

• Focus on client satisfaction

• Embraces change

• Requirements emerge

• Efficient and simple code

Disadvantages

• Requires experience of ceremonies

• Requires teamwork skills

• Giant “TODO” list lacks design overview

Compare SDLCs

choose Formal Models

• Customer knows what they want at the start

• Stable, precise and known requirements

• Change is not expected

• Mature technologies and tools

choose Agile Models

• Customer gives time to project

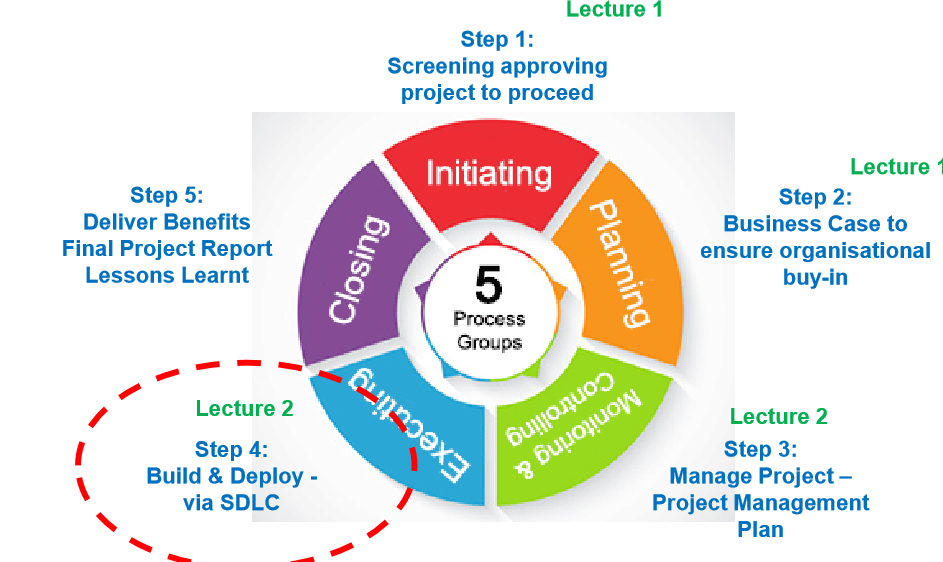
• Requirements continue to emerge

• Change is welcome

choose a hybrid model

• Client has a prescriptive model established

SDLC is a Process



**Module 7:**

**Risk**

A risk is a:

Possible future event that has negative results

Hazard; peril; or exposure to loss or injury

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

•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 vs Uncertainty

•Risk is different to uncertainty although they are related.

•Uncertainty:

–Lack of complete certainty about an event/outcome

–The event/outcome has a probability oflessthan1 –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.

Why formal Risk Management

•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

**Risk Management Process**

•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?

Risk Management Planning

•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

Risk Management Plan Template:

•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

Characteristics of Risk

•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

**Kinds of Risk**:

•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.

•Rick identification techniques

–Pondering

–Interviewing

–Brainstorming (risk framework, Work Breakdown Structure(WBS))

–Checklists

–Delphi Technique

–SWOT Analysis (Strengths, Weaknesses, Opportunities and Threats)

(P19-21)

**Risks Analysis and Assessment**

•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

**Risk Analysis**

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 theproject

-Usually measured in a scale of 1–5(or10):

(1)no impact;(2)minimal impact;(3)moderate impact;(4)severe impact (5)catastrophic impact

-Impact can be expressed as a monitory value

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

4.Identifying the root cause

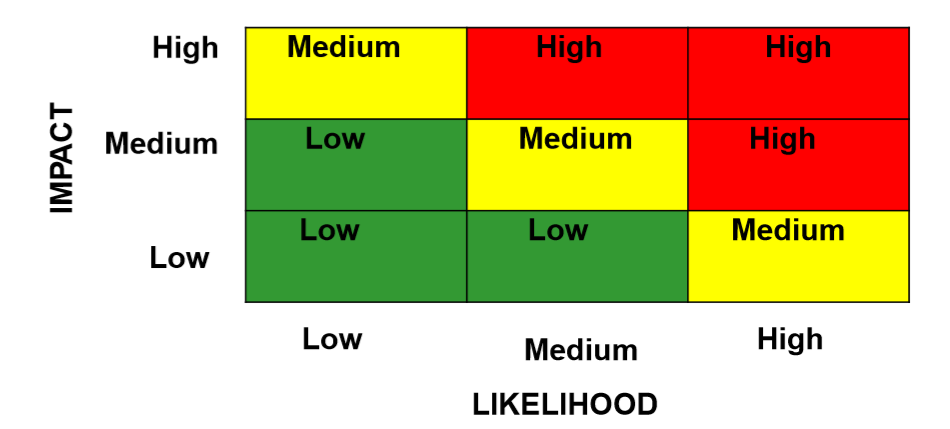
-It is important that one identifies the root causes of all risks

-If this root cause can be identified, then all of these risks can be controlled by addressing the root cause

**Risk Assessment –Risk Matrix**

•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



Risk Assessment -Quantitative

•Common Techniques:

–Decision Tree Analysis

–Simulation

–Sensitivity Analysis

**Respond to Risks (strategies)**

•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.

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.

**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.

**Risk Response Plan (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

**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)

**T3:**

RISK (do in initiation phase)

Example of risk register

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

Respond – mitigate Risk in Sprint

Monitor – during Sprint Review & Retrospective, Sprint 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

• 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)

**Module 8:**

**Formal Project Scheduling**

Challenged Projects – why?

1. Lack of a Scope Document

2. Inconsistent Communication

3. Unrealistic Expectations and Deadlines

4. Incompetent Project Manager and Team

5. Lack of cohesion between team members

6. Poor Monitoring and Risk Management

7. Poor Planning

**Project Planning:**

– Project Scheduling (Lecture 4)

– Cost Estimation (Lecture 6)

– Risk Management (Lecture 3)

– Quality Management

– Configuration Management (Change Management)

– Resource Management

– Communication Management (Lecture 5)

• **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

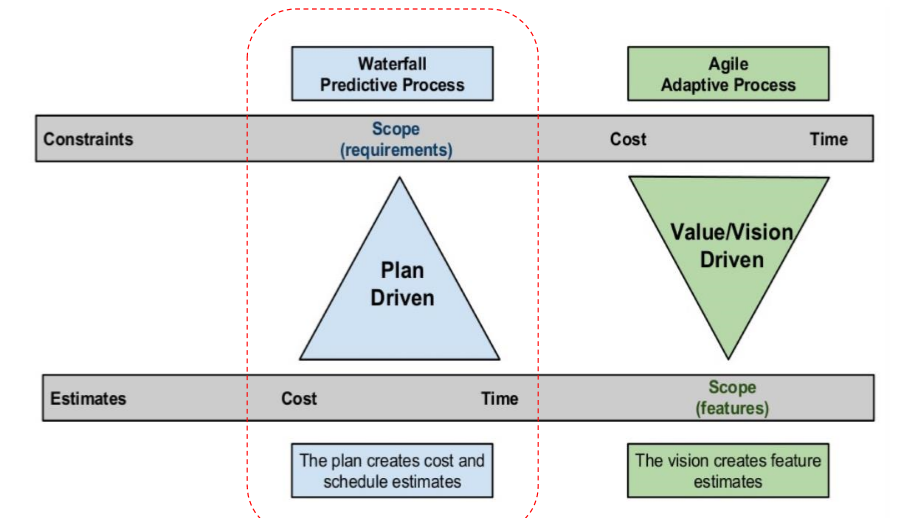
– People and physical resources required by each task

– Milestones and deliverables

– Project Timeline

formal SDLC processes – Plan Driven

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



**Steps to Develop 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

Work Breakdown Structure (P32)

• 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)

Identify Task Dependencies

• Tasks can be: – Unconstrained: the task can start at any time (buy paint, remove detachable decorations) – Constrained: depends on another task (cannot remove wall paper until decorations are removed)

• If task B depends on task A (A ->B)

– B is a Successor task (S)

– A is a Predecessor task (P)

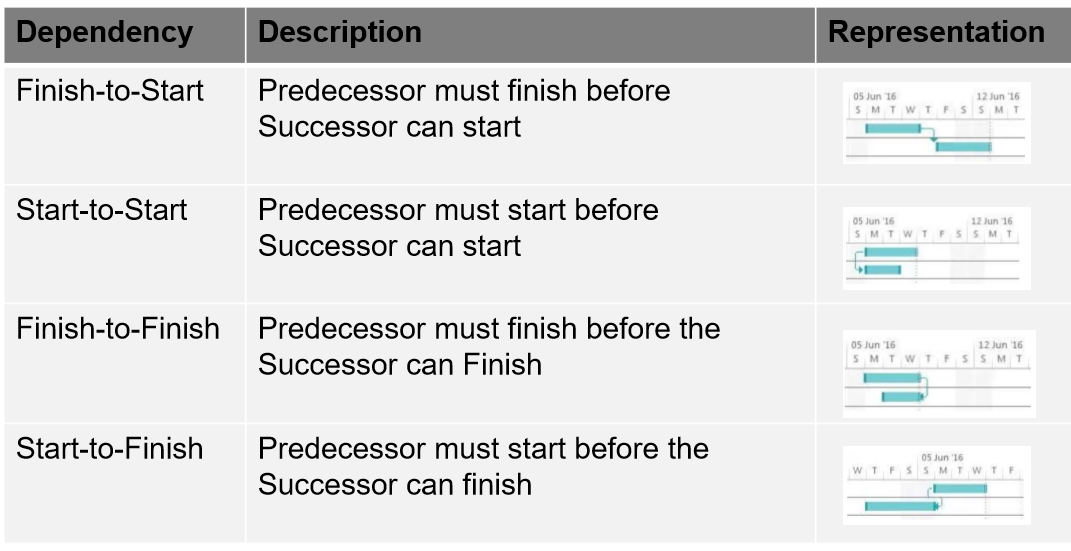
• Remove Detachable Decorations (P) -> Remove wall paper (S)

• Dependencies are caused by:

– a task needing a work product of another task

– a task needing resources used by another task

Types of Task Dependencies



Effort-time Estimation

• A common measure for estimating the effort for software is man-months (more generally person-months)

• 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

Time Estimation

• Terminology

optimistic time O

pessimistic time P

most likely time M

expected time - 𝑇𝐸

𝑇𝐸 = Τ (𝑂 + 4𝑀 + 𝑃)/ 6

Resource Allocation

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

𝑁 =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

**Project Schedule**

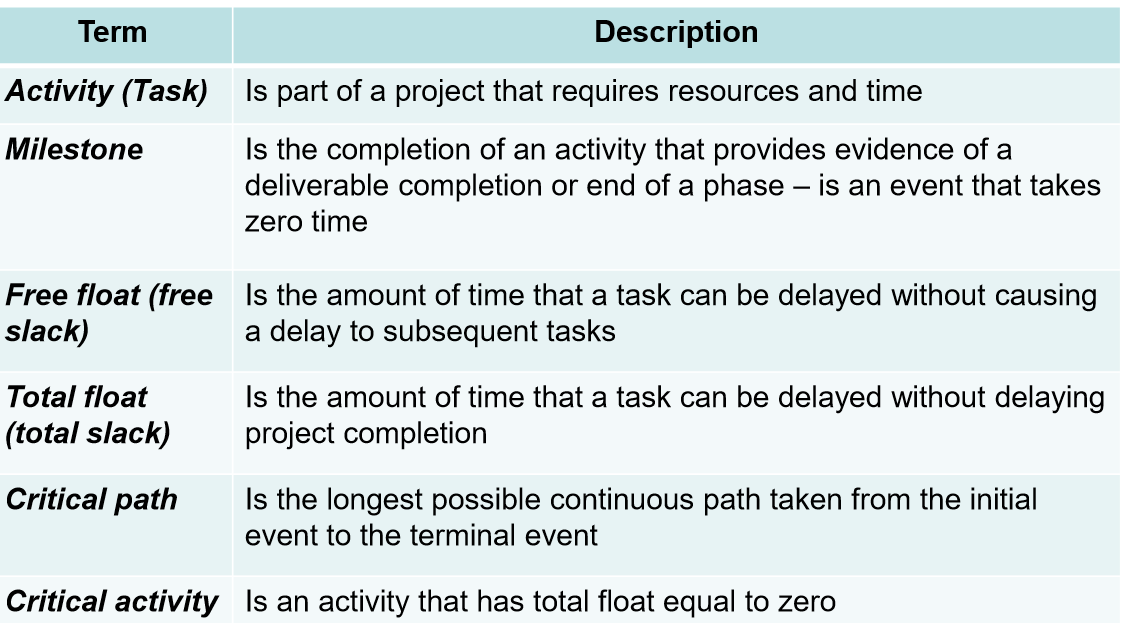
• Project Schedule will answer two important questions not answered so far: – How long will the system take to develop? – How much will it cost?

• 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

Definition of project scheduling



• 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

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 Chart

• 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:

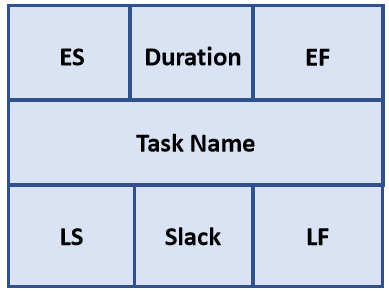
• Earliest start time (ES)

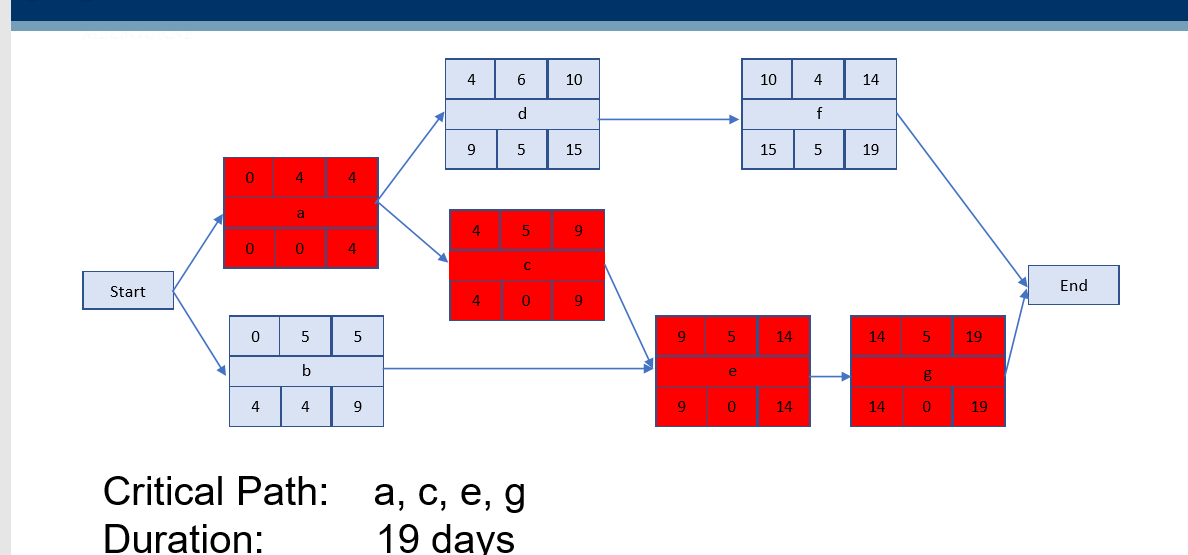
• Latest start time (LS)

• Earliest finish time (EF)

• Latest finish time (LF)

• Slack time





Notes: • Critical path activities have a total free slack of 0

• Two parallel paths could be critical paths

Critical Path Methods

• 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

Project Tracking and Control

• How to track and control project progress?

– Periodic meetings where team members report progress

– Evaluating the results of reviews and audits conducted as part of the software engineering process

– Tracking formal project milestones

– Comparing actual start dates with scheduled start dates

– Meeting engineers and having informal discussions

– Using a formal method like earned value analysis

Earned Value Analysis (EVA)

• 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

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

• Schedule Variance: expressed in dollars SV = EV - PV = 20,000 – 25,000 = (5000)

• Schedule Performance Index: expressed as a fraction SPI = EV/PV = 20,000/25,000 = 0.

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

• Cost Variance: expressed in dollars CV = EV - AC = 20,000 – 35,000 = (15,000)

• Cost Performance Index: expressed as a fraction CPI = EV/AC = 20,000/35,000 = 0.57

**Module 9:**

**Planning in Agile Development**

• Takes a significantly different flavour from traditional approaches

• Detailed planning is deferred until the start of the iteration – Designed to handle change – An iteration includes all phases (requirements, design and test)

• Planning is based on light weight lists – Gantt and PERT charts are considered less useful

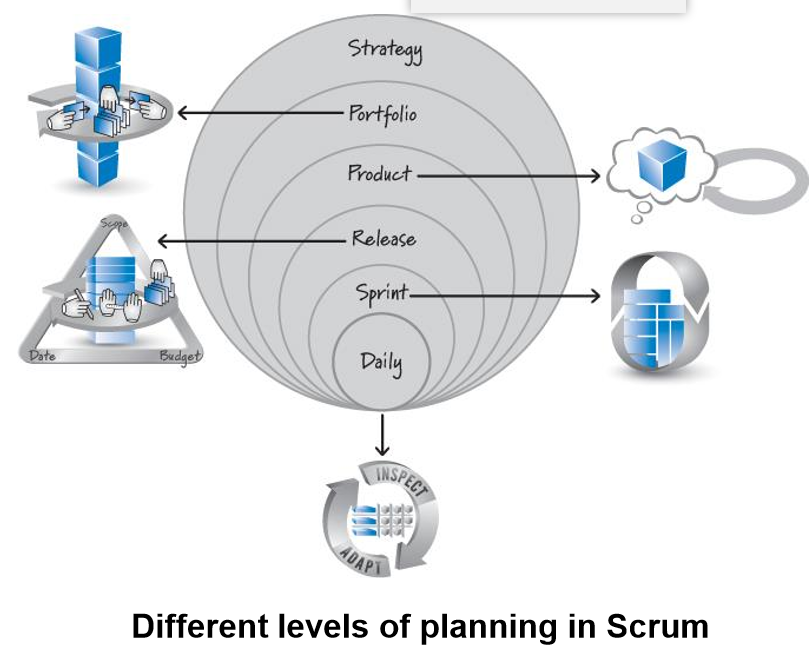
• Plan short iterations

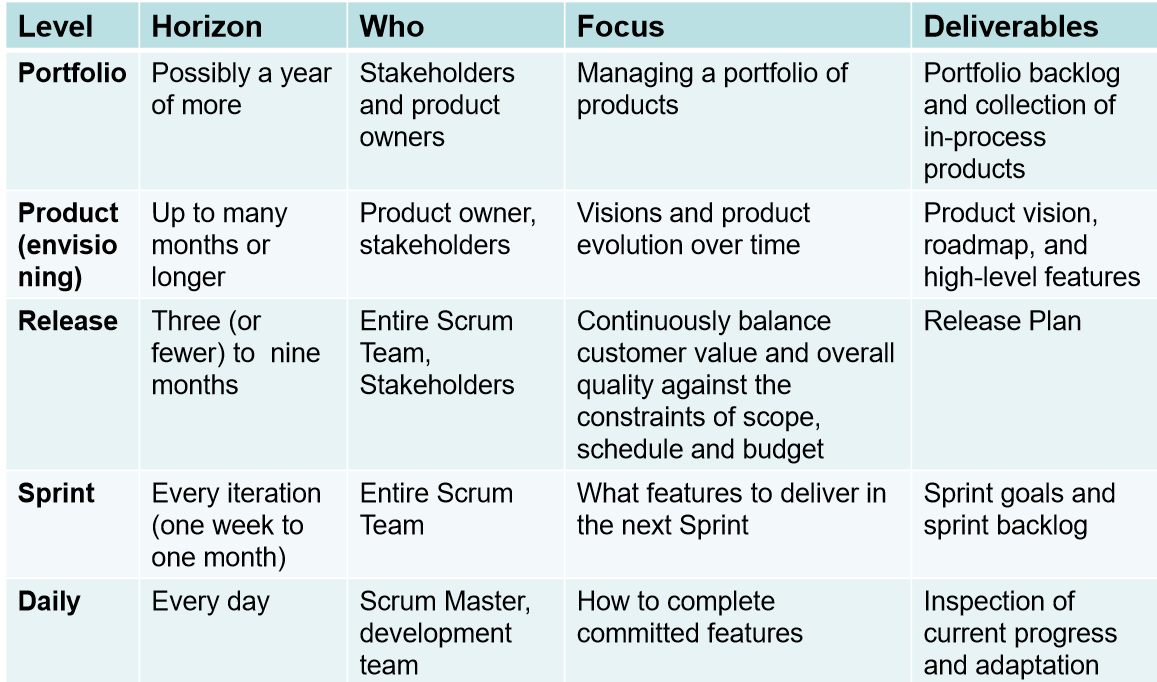
• Deliver working software

• Use “Just in time (JIT) planning” – next iteration

• Use the team

**Planning in Scrum**





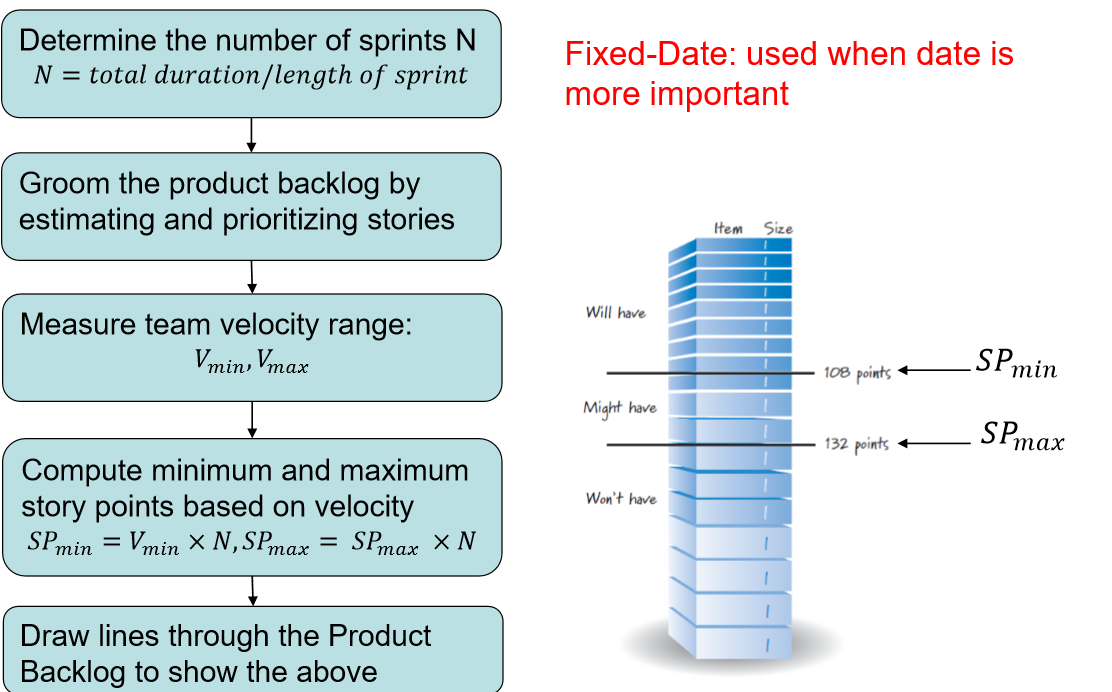
• Agile Planning

– Recognizes that all three factors: scope, budget and time cannot be fixed in reality - not recommended

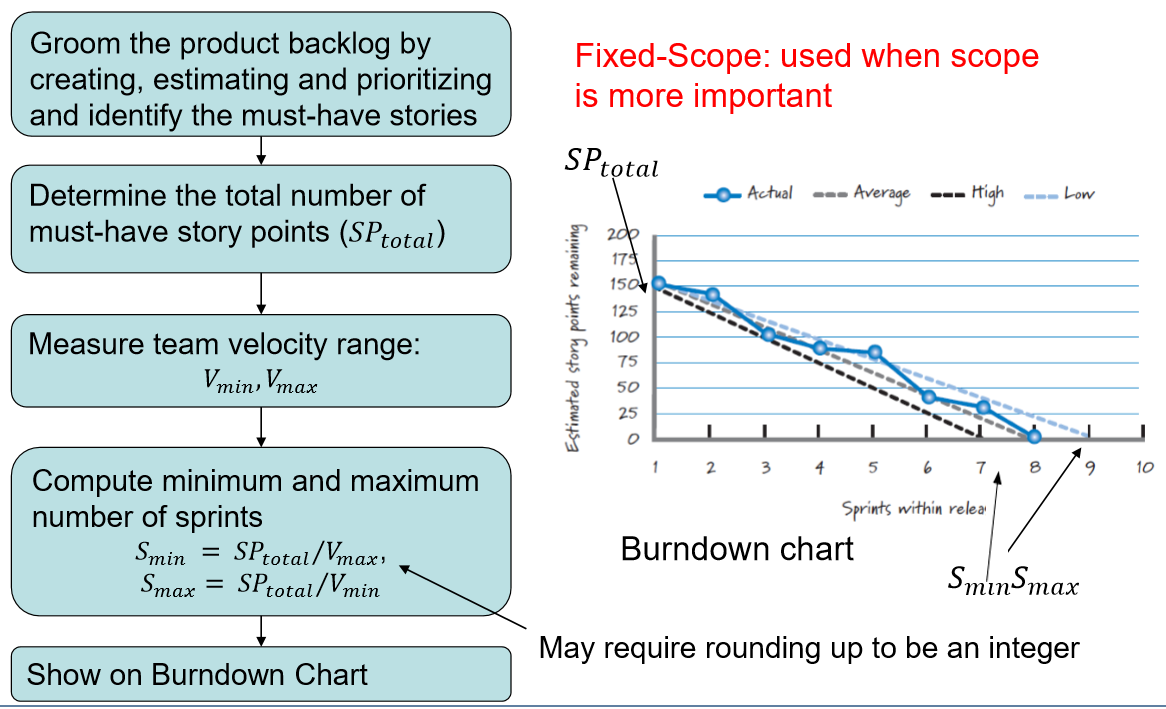
– Can we fix scope and date and make the budget flexible?

• Not really because increasing the budget, hence the resources will not always help to improve speed – not recommended

**Fixed-Date release planning**



**Fixed-Scope release planning**



• Assumptions in Formal Planning:

– Scope fixed – requirements are stable

– Budget fixed – cost estimations are accurate

– Schedule fixed - derived based on scope and budget

**Module 9-11**

**Module 9 – Individuals, Motivation and Teams**

**Organisational Theory & Motivation**

Maslow Hierarchy of Needs:(P16)

• 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

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

• Remedying the causes of dissatisfaction will not create satisfaction

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

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

**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.

Team definition:

• 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

Benefits of team:

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.

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.

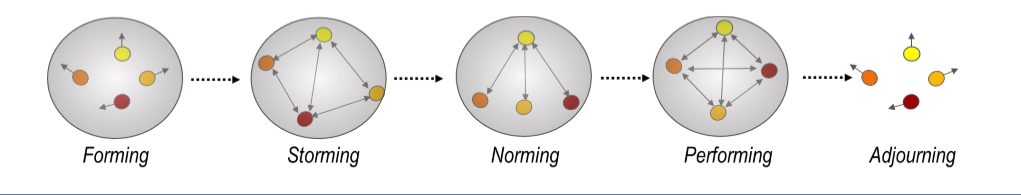
8. Increased Equality: Individuals across all levels can more freely offer their ideas, knowledge and concerns.

9. More Dialogue: Teams offer a site where people can voice their feelings, disagreements, opinions and ideas.

teams Form & Perform

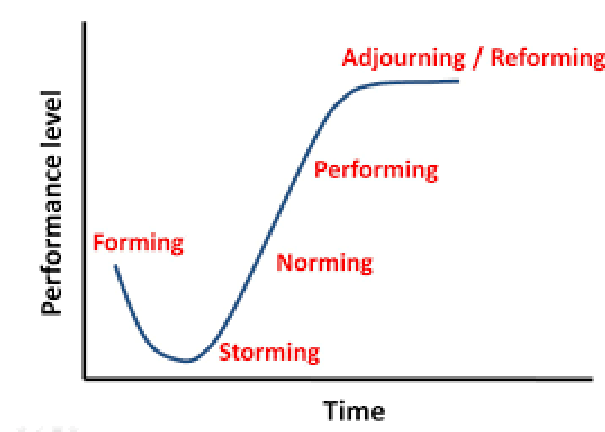
Tuckman’s Team Development Model

• 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

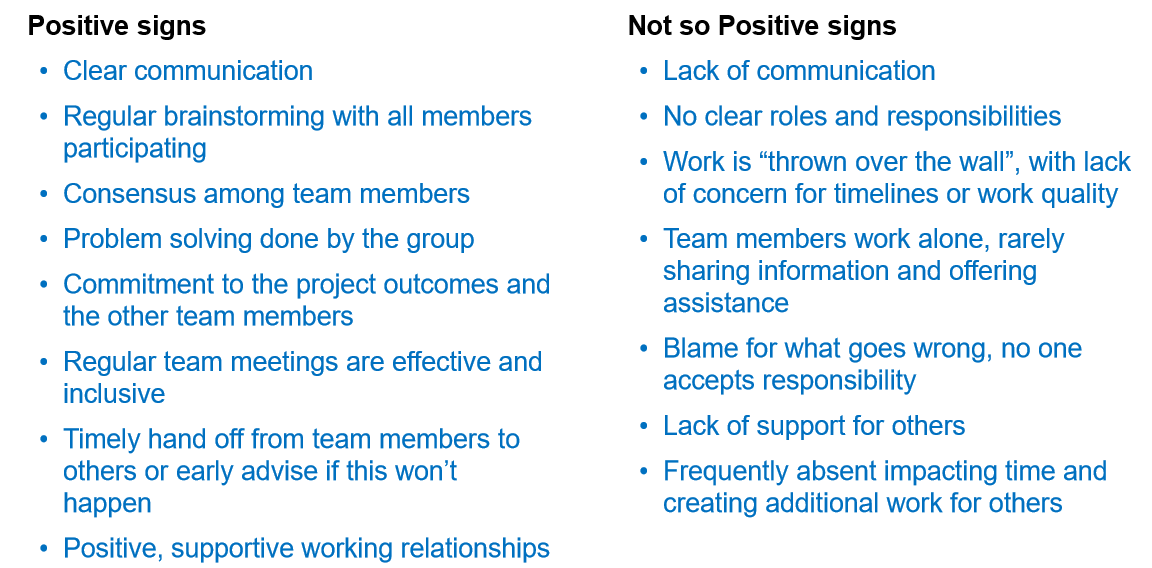


• As teams develop maturity and ability – leadership styles change and behaviors change

• Tuckman also found that Team Effectiveness changed over time with the team experiencing initial decline in performance after Stage 1 – Forming



Team Effective:



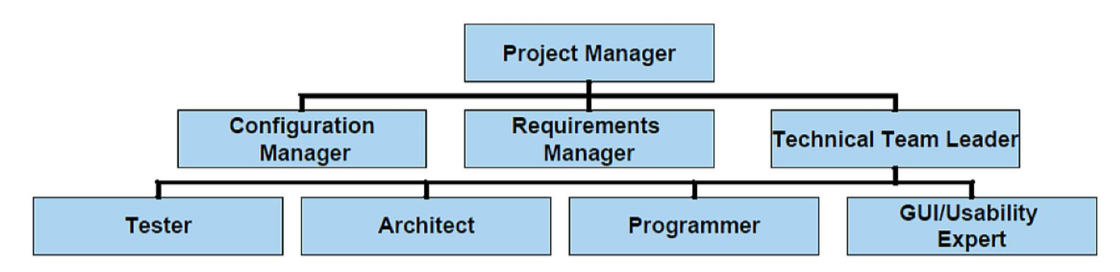
**Team Structures**

Controlled Centralised

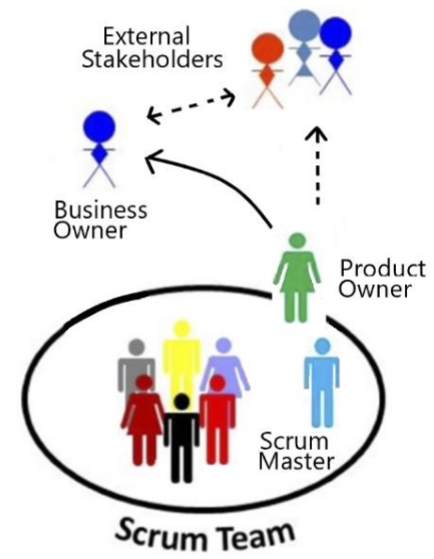
• Leader coordinates tasks and directs work

• Communication and Control are vertical

• Sub-teams with leaders to direct and guide sub-groups



Scrum Team



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

**Module 10 – Communication Management**

Communication Challenges(P41)

The importance of Listening

Why Do We Listen?

• Promotes problem-solving abilities

• Demonstrates acceptance of others

• Builds and retains trust in relationships

• Increase speaker’s receptiveness to thoughts and ideas of others

• Increases self-esteem of the speaker – someone cares

• Helps you understand and retain information

• Allows you to help others

The Process of Listening

• Predicting [some expected outcome]

• Receiving

• Assigning meaning

• Assess / Validate

• Remembering

• Types of Listening

– Passive Listening – Lectures / Presentaitons

• Taking in the information with little processing or reacting

– Active or Empathetic Listening – Tutorials / Team work

• Show interest • Asks questions • Avoid distractions • Use direct eye contact • Do not interrupt • Read both verbal and nonverbal messages

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

The Importance of Active Listening

• Shows the speaker you are 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

**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

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

**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

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

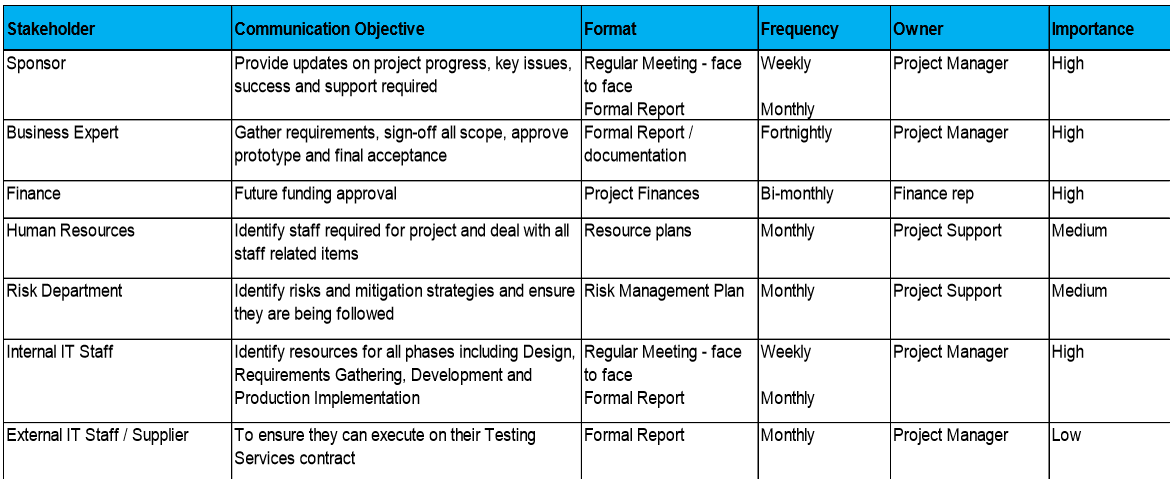
• How sensitive or confidential information will be communicated & who will authorise this

• 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

Communications Matrix

**Virtual Teams & Communication**

• 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.

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

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

• You must implement good practices to ensure everyone is involved!

Leverage communication technologies

Team building – more important in a virtual world

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

**Key Communication Consideration**

Importance of Face to Face meetings(P70)

Items to remember

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 11 – Stakeholder Management**

Identifying Stakeholders & the Stakeholder Register

**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

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

Stakeholder Analysis includes:

• Names and Organisations of Key Stakeholders

• Their Role on the Project

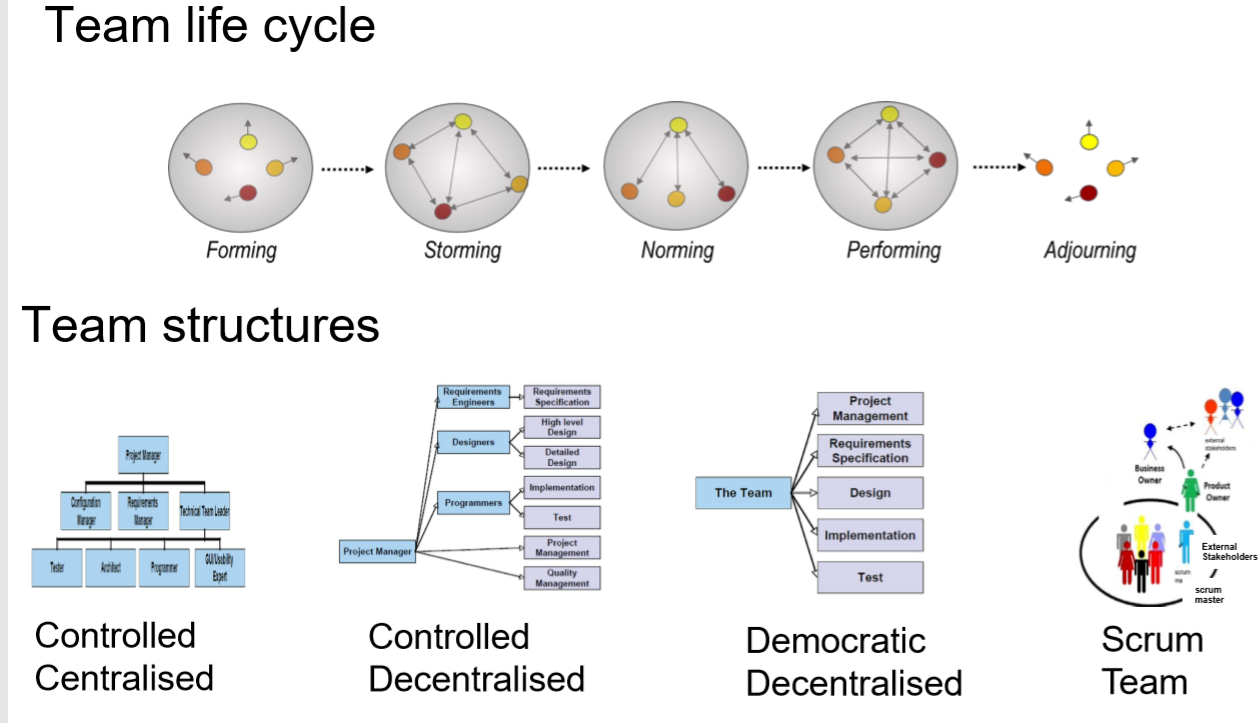
• Unique Facts about Each Stakeholder

• Level of Interest in the Project

• Influence on the Project

• Suggestions and Strategies for Managing Relationships with each Stakeholder

**T6: Team structures**



Team roles:

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

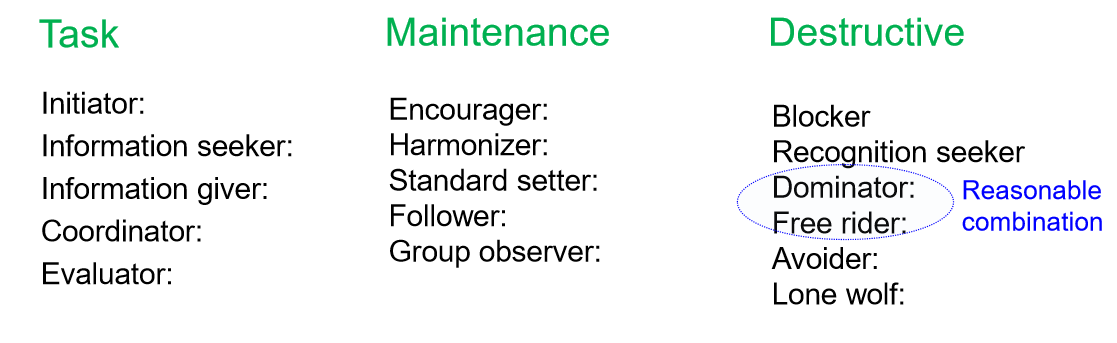
Encourager: praising, accepting, cohesion and warmth

Harmonizer: build consensus, humor to neutralize anger

Standard setter: focus on goals, standards

Follower: agreeable

Group observer: provides feedback



**Virtual team**

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

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

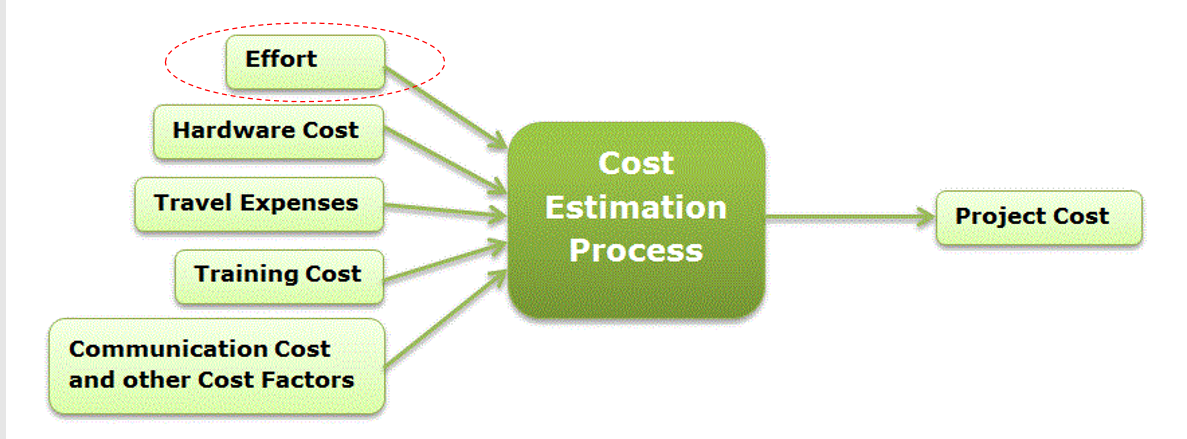
• (very similar to factors that contribute to any successful team)

S.M.A.R.T. Feedback



**Module 12-13:**

determines project cost:



• 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

Challenges in cost estimation:

• 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

Possible solutions；

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

**Techniques for 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 = Τ (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

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 × 𝑆𝑖𝑧𝑒𝐵 × 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

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

**Software Size 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

Source Lines of Code (SLOC)

• 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

• 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

• 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

Function Points (FP)

• 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)

Software Requirements Specification (SRS)

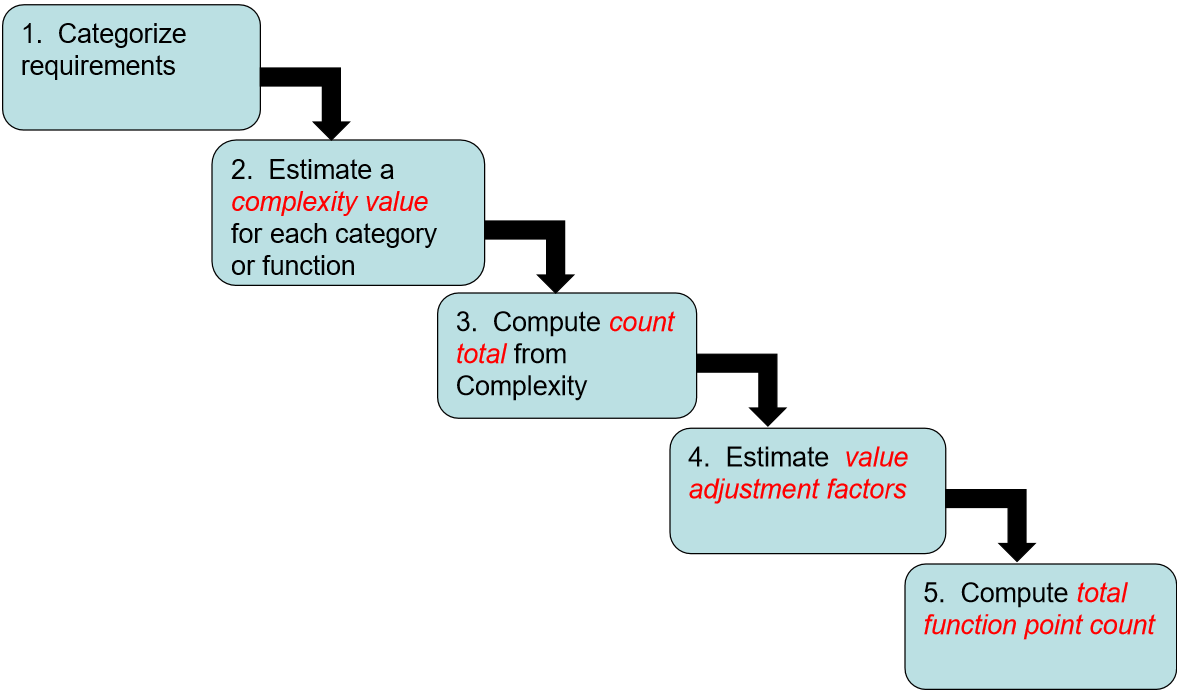
– A document that specifies what is expected of a software system; referred to as the requirements of the system

– It contains:

• Functional Requirements: – Specify the functions that are required in the system

• Non-functional Requirements: – Specify requirements that are not directly functions, such as performance, reliability, scalability etc. (quality requirements)

FP Computation – Step



1. Categorize Requirements

Data Functions: • Concerned with maintenance of the data for the application – Internal Logical Files (ILF), External Interface Files (EIF)

Transaction Functions: • Concerned with information being passes to and from the system External Inputs (EI), External Output (EO), External Inquiries/Queries (EQ)

five categories:

– Internal Logical File (ILF) • A logical grouping of data that the system maintains over a period of time, and is modified using external inputs examples - tables in a relational database, files containing user setting

– External Interface File (EIF) • A logical grouping of data that is maintained external to the system, but which may be used by the system. examples - are the same as ILFs, except that the data is maintained outside the system, such as data hosted on a thirdparty servers, or data structures holding information about system state

– External Input (EI) • An input to the system from a user or another application, which is used to control the flow of the system, or provide data. External inputs generally modify internal logic files examples - data fields populated by users, inputs files (e.g. program source code to a compiler), and file feeds from an external application.

– External outputs (EO) • An output to the user that provides information about the state of the system examples - screens, error messages, and reports that are shown to the user. Individual data fields in these are grouped as one external output

– External Inquiries/Queries (EQ) • the input is not used to update an internal logic file, but is used to query the internal logic file and provide an output; the output is retrieved directly, with no derived data included examples - reading a user setting, or reading a record from a database table

2. Estimate a complexity value for each category or function

– Complexity is ranked either simple, average or complex

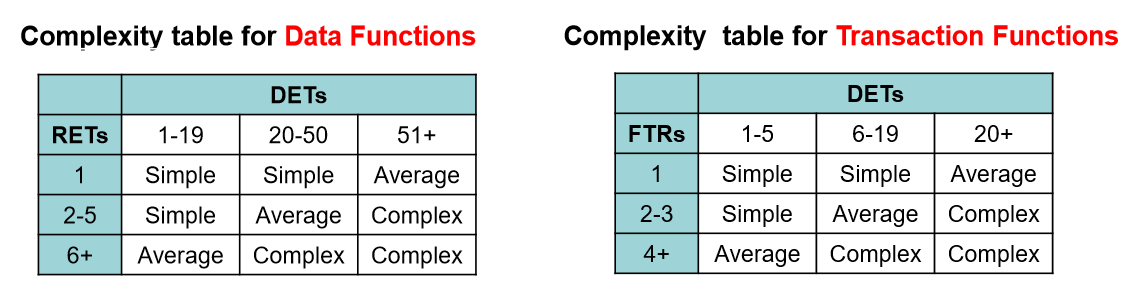
– Normally assigned for a category rather than for each requirement – rather crude

– A technique commonly used is based on Data Element Types (DETs), Record Element Types (RETs), and File Type References (FTRs):

Data Element Types (DETs) A unique, user-recognizable, non-repeated data field in a system

Record Element Types (RETs) A user-recognizable subgroup of data elements in an ILF or EIF

File Type References (FTRs) A file (ILF, EIF) referenced by a transaction

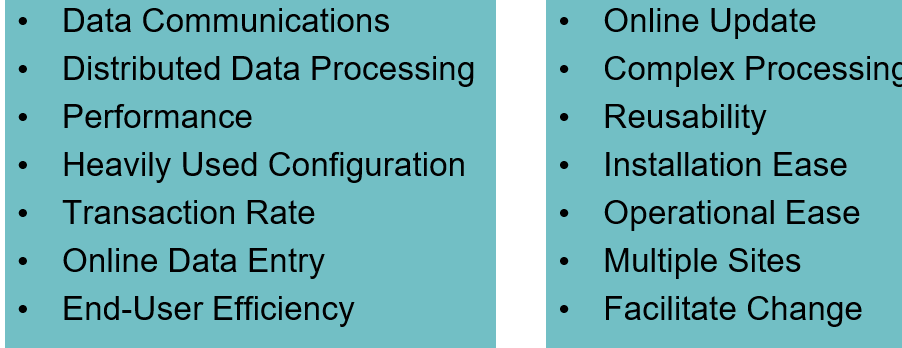


3. Compute count total from Complexity

• Using count and complexity estimates compute total count

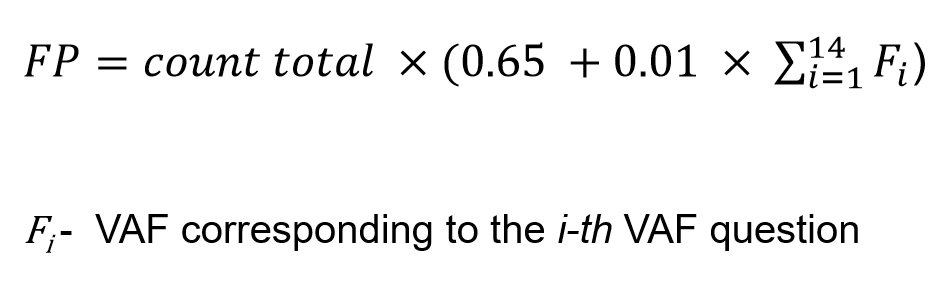
4. Compute value adjustment factors

• is computed based on 14 characteristics

• each of the characteristics is ranked on a scale of 0-5; 0 not important and 5 critical 

5. Compute total function point

• Count total and value adjustment factors are then plugged-in to the following formula to estimate the total point count



**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

**Use Case Points**

• 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

• 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

Step:

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

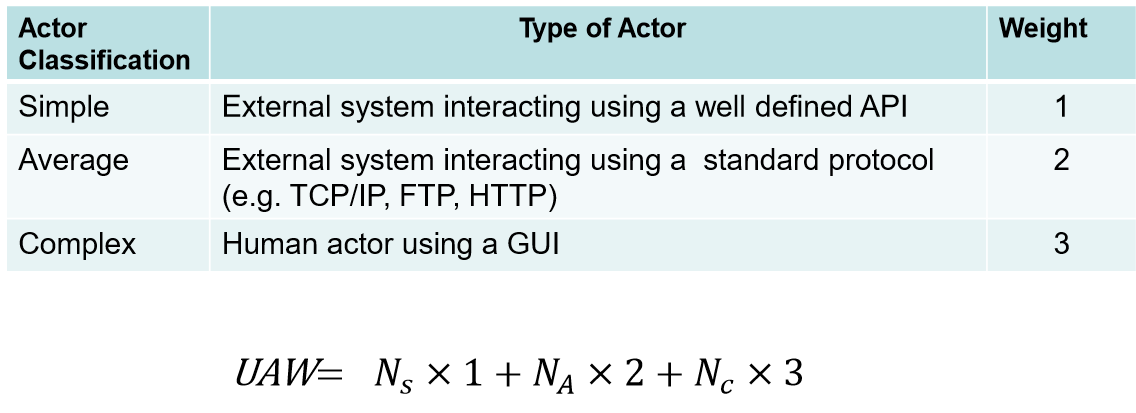
1. Compute Unadjusted Use Case Weight (UUCW)

Count the number of simple average, complex use cases, Ns, NA, NC based on the number of transactions as per table below.

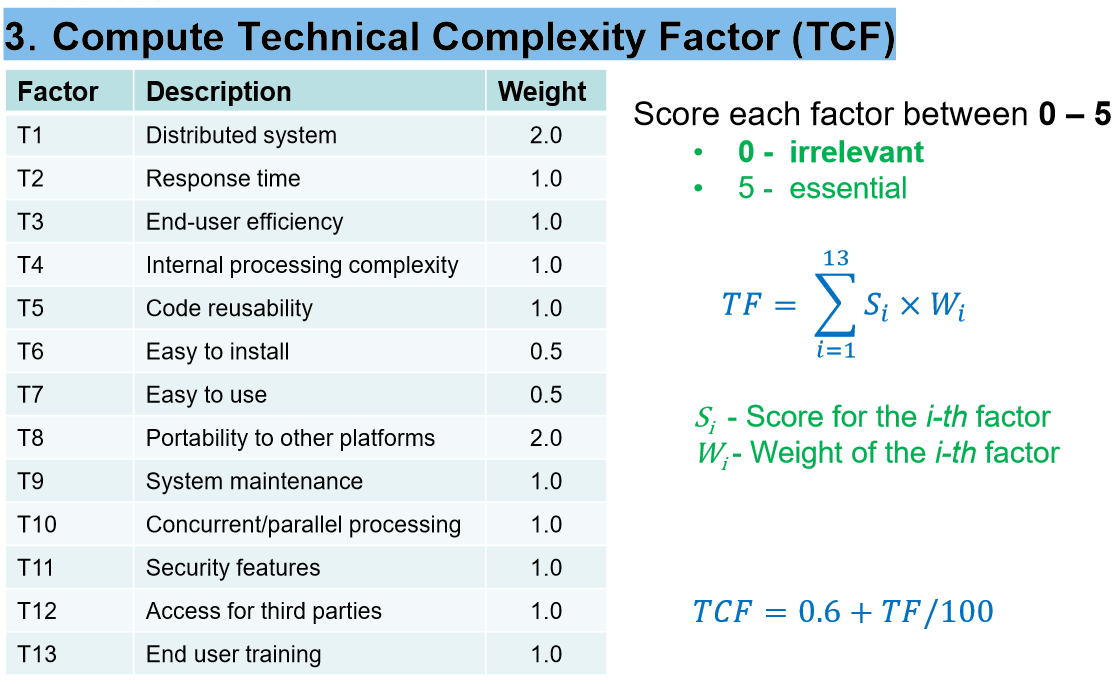
– The number of transactions can be computed by counting the number of steps in the scenario



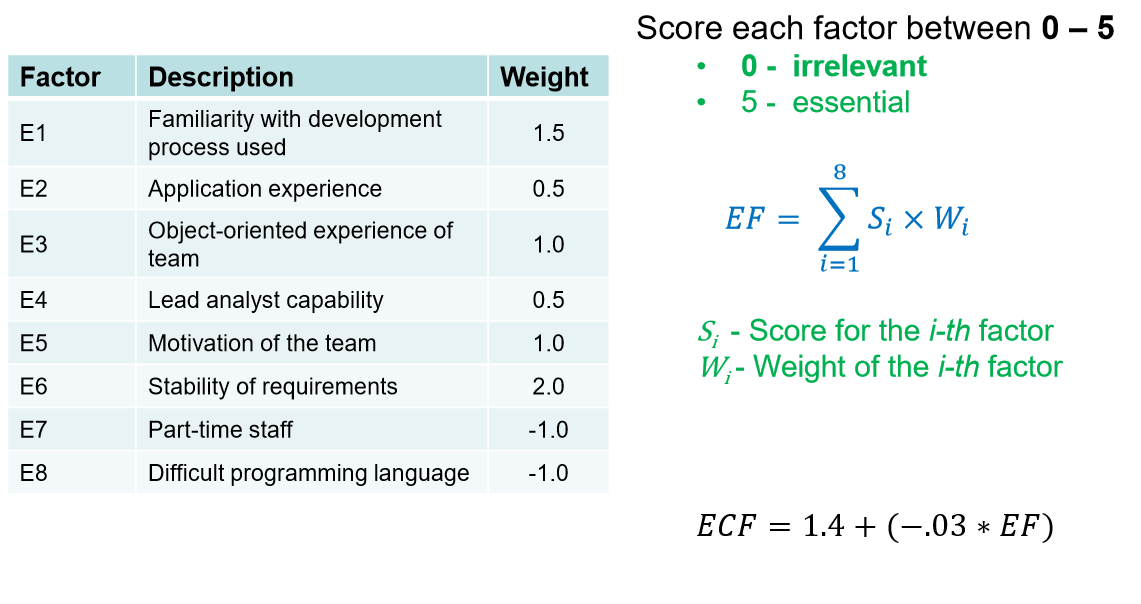
2. Compute Unadjusted Actor Weight (UAW) Count the number of simple average, complex actors, Ns, NA, NC as per table below.



3. Compute Technical Complexity Factor (TCF)



4. Compute Environmental Complexity Factor (ECF)



5. Compute the final size estimate

𝑈𝐶𝑃 = (𝑈𝑈𝐶𝑊 + 𝑈𝐴𝑊) × 𝑇𝐶𝐹 × 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

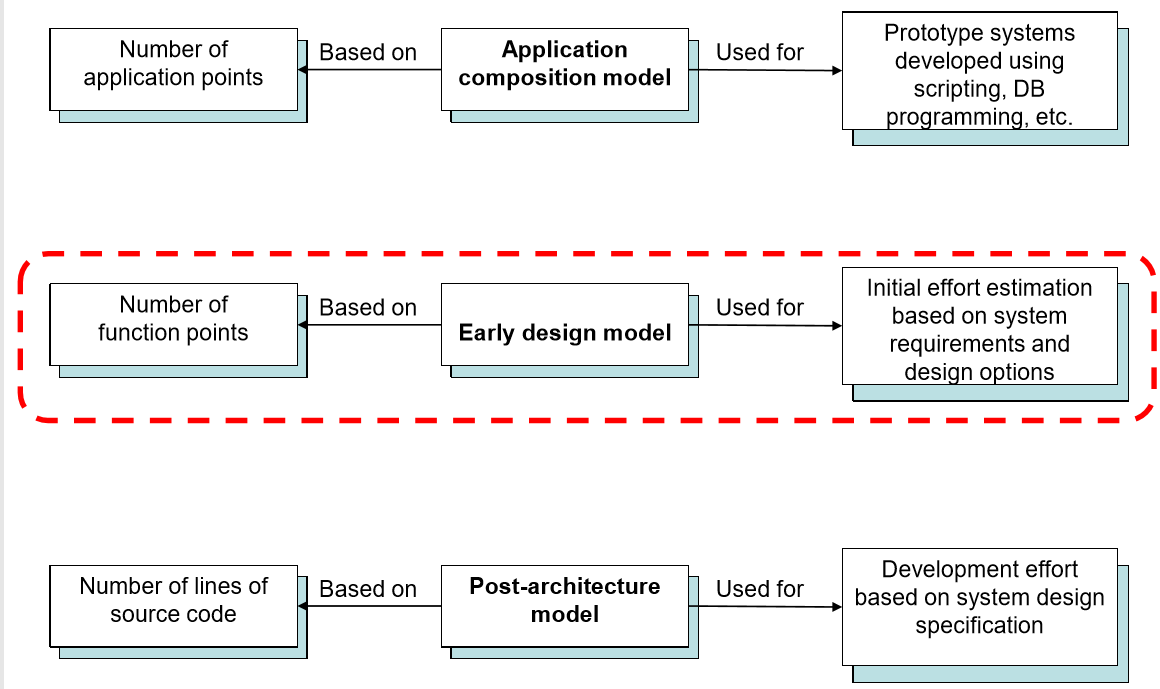
**COCOMO Model**

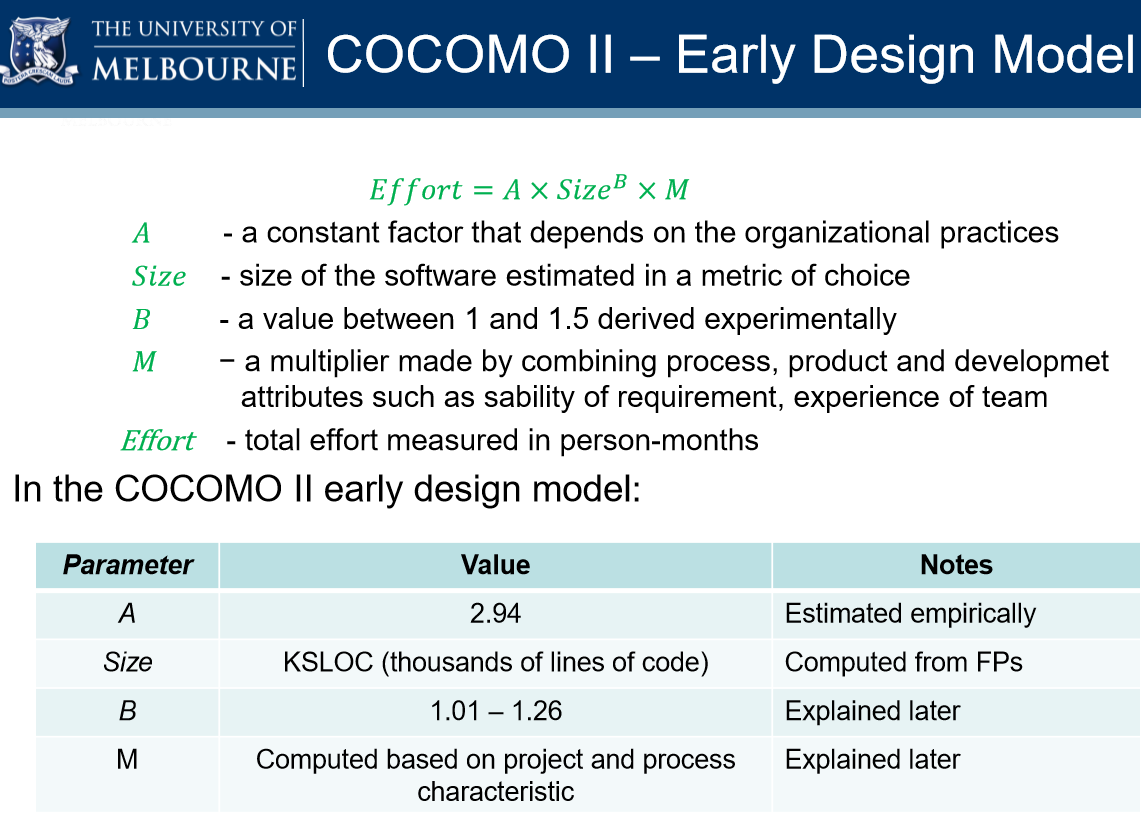
• Derived from collecting data from a large number of software projects and deriving formulae that best fits the observations – an empirical model

• 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

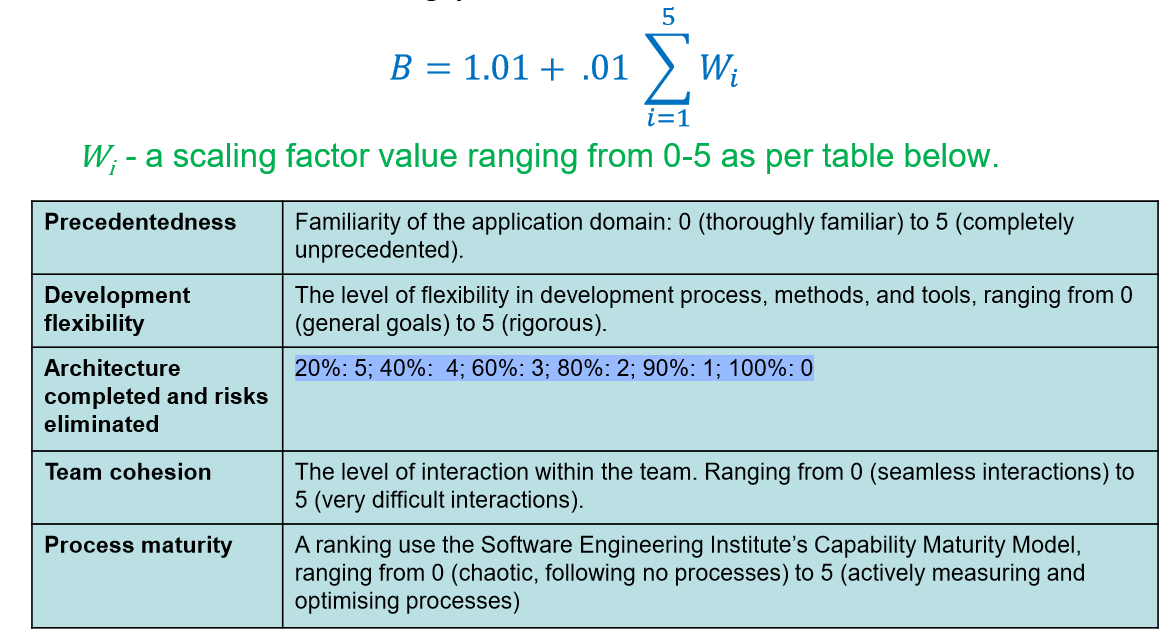




COCOMO II: Estimating Size

in KSLOC (𝐸𝑓𝑓𝑜𝑟𝑡 = 𝐴 × 𝑺𝒊𝒛𝒆𝐵 × 𝑀) – Size is estimated based on logical lines of code – Can be estimated based on FPs using the table below (P76)

Estimating parameter B

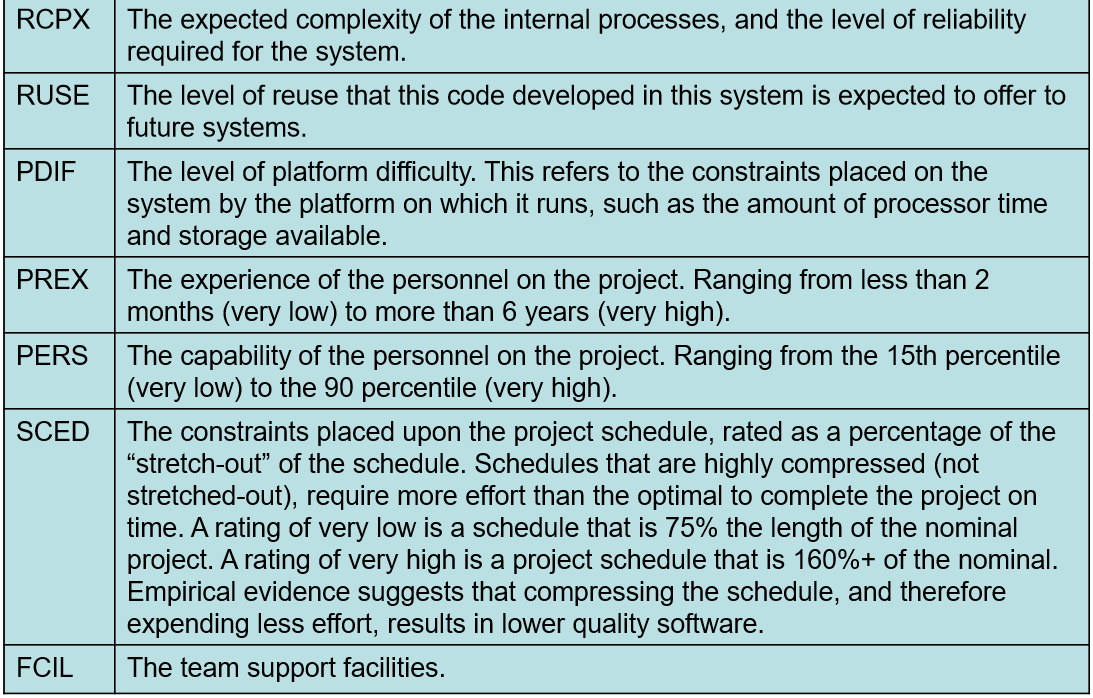


Estimating M

– the cost drivers consist of seven different factors

– each factor is rated on a six point scale; very low to extra high

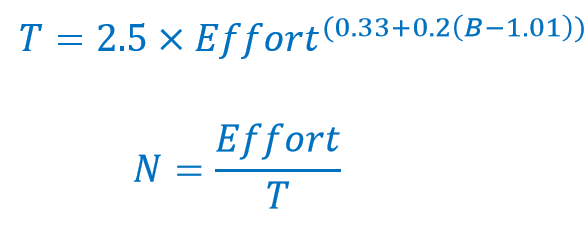




COCOMO II:

Estimating Time ( T ) and Number of Personnel ( N )

- Formula for estimating the nominal delivery time:



**Module 13 Agile Effort Estimation**

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

Agile Estimation Process:

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.

Agile Estimation Guidelines:

• 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.

**Agile Estimation Techniques:**

– Planning Poker

– Bucket System

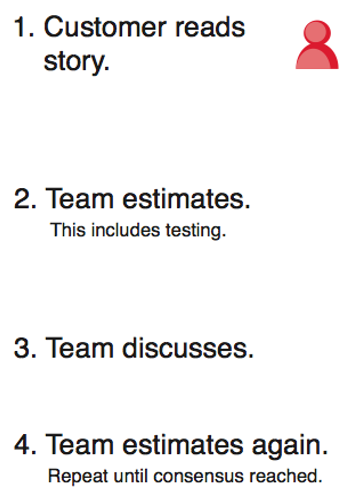
– Relative Mass Valuation

– T-Shirt Sizes

– Affinity Estimation

– Dot Voting

Planning Poker:



BUCKET SYSTEM:

• The team sitting at a table picks a user story card randomly and places is 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 an appropriate bucket, based on individual judgement (Divide and conquer)

• Finally the team reviews the placements and reach agreement

Relative Mass Valuation

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.

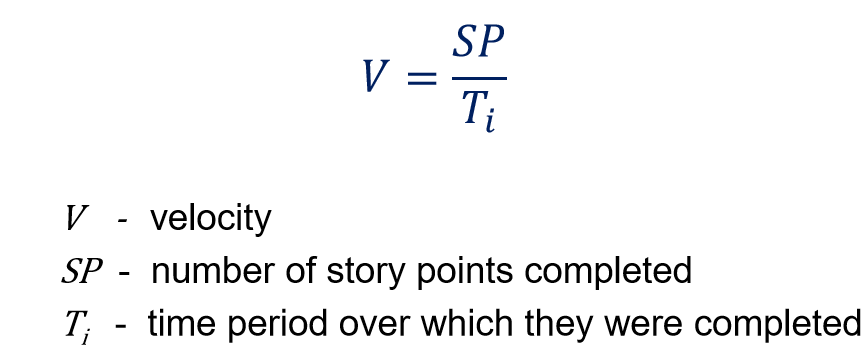
3. Large story one end on the table. Medium story in the middle and Small story the other end

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.

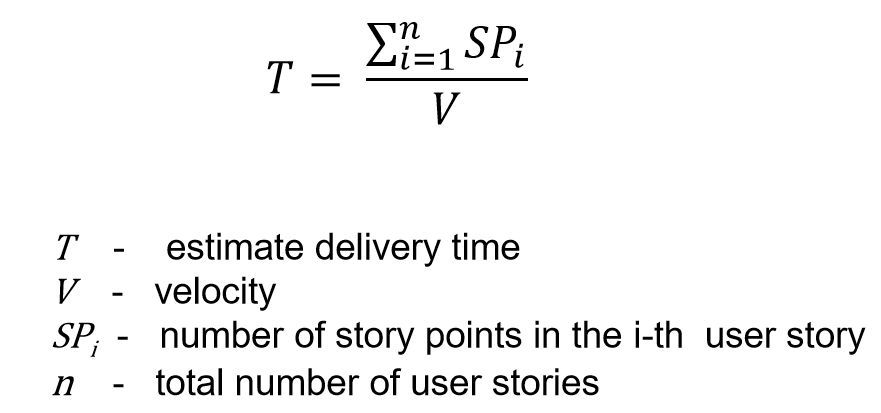
Computing Velocity



Common methods for measuring velocity:

– Using historical data – Using data from previous iterations

Estimated Delivery Time



**Final comments** on estimation

• 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

**T7:**

Cost estimation

Agile: User Stories and Story Points and Velocity

Formal: Function Point Analysis and COCOMO II

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

(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

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

raw values are unimportant

relative values matter

Scrum Release Planning (Initiation)

• Business Roadmap identifies candidate project

• Product vision established with external stakeholders

• Create Product Backlog

• 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 (every phase)

• Create Sprint Backlog

• 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

The User Stories can be decomposed into tasks,

• Optionally estimate tasks in hours (less accurate)

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

Sprint Monitoring (sprint phase)

• Sprint Burn-down chart monitors actual velocity • Scrum Master updated chart after daily stand-up

Agile Scrum Velocity

• The Scrum Master can track remaining effort

• Predict when the release milestones will be reached

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

Formal Function Point Analysis and COCOMO II

**Module 14:**

**Quality Management**

• 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 build quality into the software through a range of Quality Management activities

software quality:

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.

Cost of quality

• 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

**Quality Management Process**

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

Quality Assurance

• Quality assurance process is primarily concerned with defining or selecting the quality standards

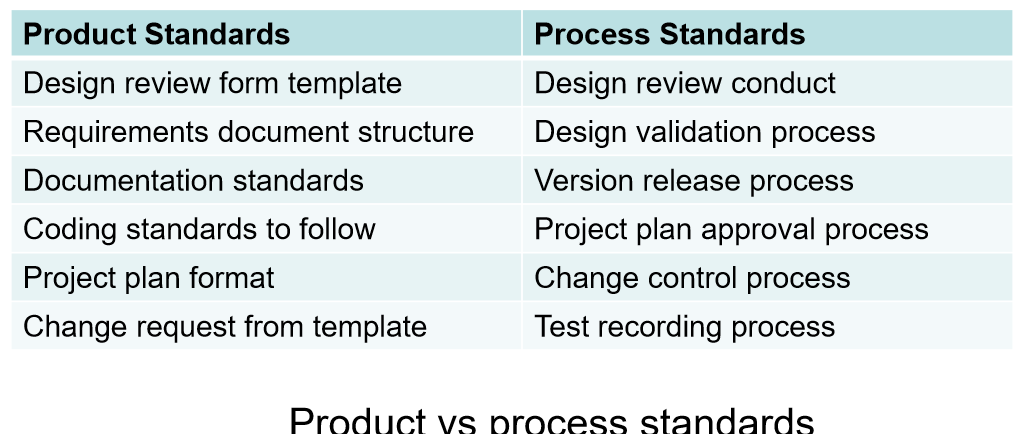
– A standard might simply be defined as a set of rules for ensuring quality

– 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



Documentation 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

Software Standards and Systems

• 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!

• Some examples of software standards and systems

– ISO 9000 – Capability Maturity Model

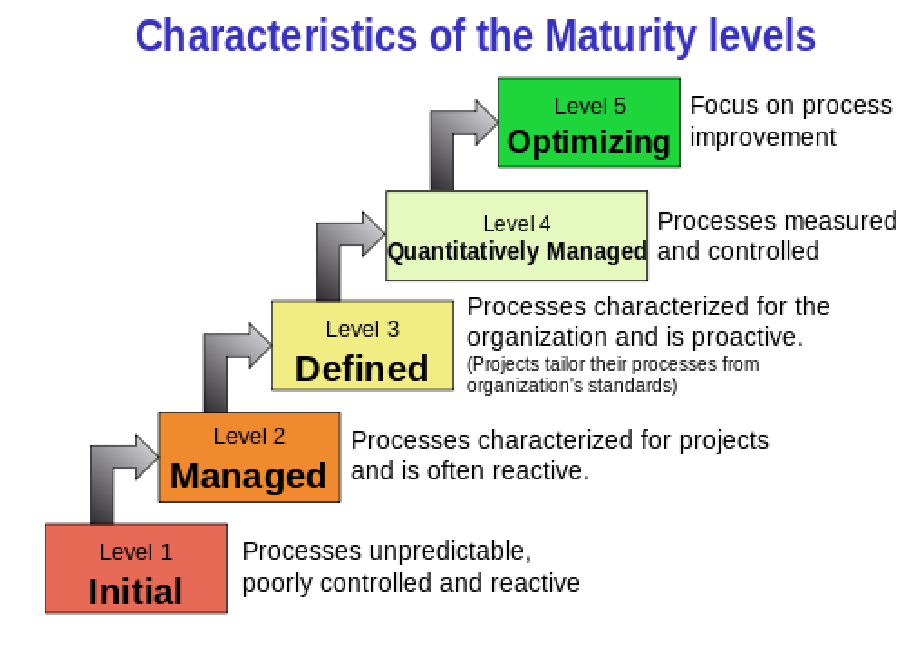
Capability Maturity Model

• 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



**Quality Planning**

• 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

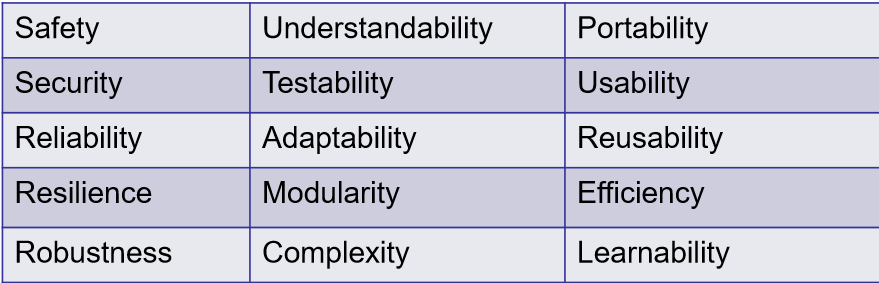
– Risks and Risk Management

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)

Software quality attributes

• Some of the quality attributes matter only matter 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



Verification vs Validation

• 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 fulfills it 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).

Reviews

• 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

Technical 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

– Process standards 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

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

**Business Reviews**

• 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

**Management Reviews**

• 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

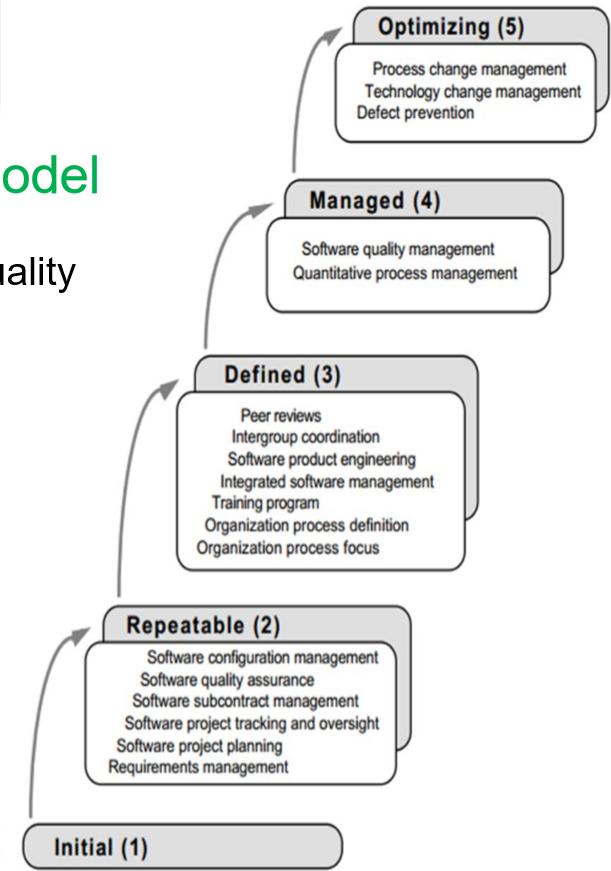
• May involve reviewing if the project meets the scope, schedule, budget and quality objectives

**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

**T8:**

Capability Maturity Model A tool for achieving high quality



Quality Plan: A checklist

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.

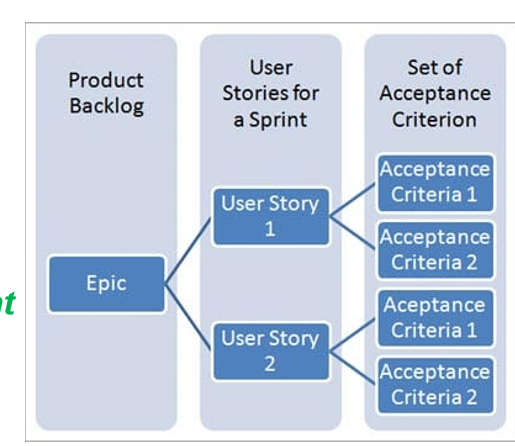
The tests can be run every time new features are released.

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

Acceptance Criteria



Sprint Review QA 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 technical debt item in the Product Backlog, (optionally in FDD format)

• The Product Owner sets the priority of the technical debt item.



**Module 15-18:**

**Module 15: Ethics and Australian Computer Society Code Of Ethics.**

Ethics:

- 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 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.

Ethics inform our day-to-day interactions

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.

**Module 16 – Outsourcing**

Definition: 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.

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

**Module 17 – Procurement**

The Procurement Management Process

If there is no need to buy (outsource) any products or services from outside the organisation, then there is no need to perform any of the procurement management processes.

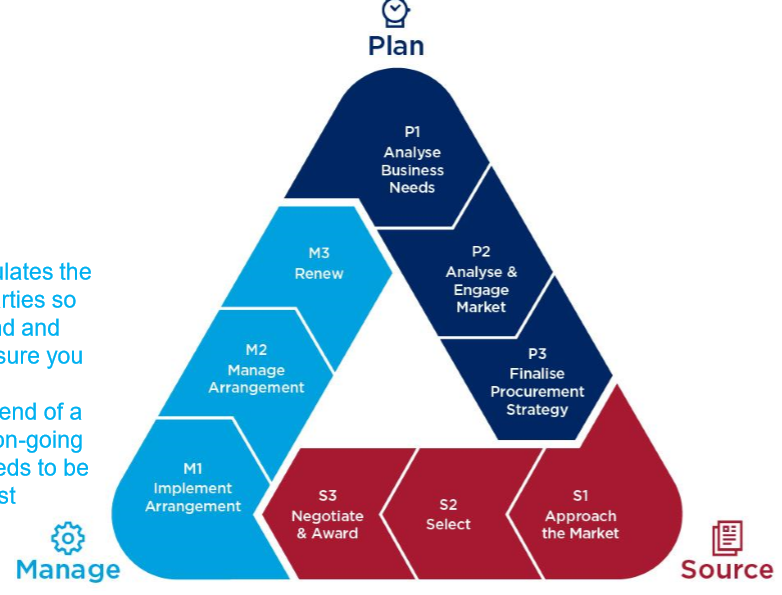
However you will find that most (if not all) projects will contain some sort of external sourcing which will require a procurement.

The Procurement Management Process consists of 3 broad stages: 1. Plan. 2. Source. 3. Manage.

**Planning** in procurement involves consulting key stakeholders to define the ‘real’ need, analysing how the supply market works, assessing risks and ultimately defining the best Procurement Strategy to meet the organisations requirements.

**Source**, the principal objective of this stage is to identify and engage suppliers who will provide the best value for money outcome, in a framework of probity and fair dealing. A key deliverable for this stage is to determine the appropriate sourcing method, with consideration given to alternatives other than just tendering.

**Manage**, Every arrangement articulates the rights and responsibilities of the parties so it is important to identify, understand and manage them in order to better ensure you ‘get what we contracted for’. Signing an arrangement is not the end of a process, but rather the start of an on-going relationship with the supplier. It needs to be managed in order to deliver the best outcome for the organisation.



Sourcing Procurements(P33)

The procurement process is typically conducted with the issuing of a Request For X (RFx), where x = Bid, Information, Proposal, Tender or Quote.

RFx:

The RFx document is prepared by the buyer and will have specific information depending on the what it is (RFI, RFP, RFT/Q). It will typically include:

1. Purpose of RFx.

2. Organisation’s Background.

3. Basic Requirements.

4. Hardware and Software Environments.

5. Description of RFx Processes & Evaluation.

6. Statement of Work and Scheduled Information.

7. Appendices:

a. Current Systems Overview.

b. Systems Requirements.

c. Volume and Size data.

d. Required Contents of Vendor’s Response to RFx. e. Sample Contract.

Statement of Work (SOW):

A key component of the RFx document is to analyse the business needs and establish a detailed Statement of Work (SOW).

A Statement Of Work is a description of the work required. A good SOW is detailed and gives bidders an understanding of buyer’s expectations, key items include:

• Scope of Work to be completed

• Location of where the Work is to be completed from

• Measurement and Performance criteria

• Deliverables, milestones and schedule

• Applicable Standards and Acceptable Criteria

• Any Special Requirements

Approach the Market, Select, Negotiate and Award:

• Deciding whom to ask and potentially do the work

• Sending appropriate documentation to potential sellers / bidders

• Obtaining proposals / bids

• Evaluating responses and selecting a preferred supplier

• Negotiating the contract

• Awarding a contract

Evaluation Processes:

1. Evaluation team review of RFx response and evaluate against predetermined criteria.

2. Schedule short-listed vendor presentations.

3. Check vendor references.

4. Short-listed vendor presentations.

5. Evaluation team site visits to short-listed vendors / references.

6. Evaluation team finalises evaluation and selects short-listed firms.

7. Best and Final Offer (BAFO) with short-listed firms.

8. Conduct final negotiation with preferred supplier.

**Managing Procurements**

Implement, Manage & Renew:

• Implement the agreement & services as per the contract and SOW

• Manage the agreement to ensures the seller’s performance meets contractual requirements

• Review and control all changes - It is critical that project managers and team members watch for Constructive Change Orders – If change is requested then contractor can legally bill the buyer for additional work

Renew / Closing Procurements:

• Involves completing, settling contracts and resolving issues

• The project team should:

– Determine if all work was completed correctly and satisfactorily

– Resolve any issues or outstanding items

– Up date records to capture all lessons learnt & outcomes

– Archive information

– Capture all knowledge and lessons learnt

• The contract itself should include requirements for formal acceptance and closure

**Module 18 – Contracts**

Contracts are the one source of truth for all activities that are to delivered by the external parties.

What is a Contract?

• A mutually binding agreement that obligates the seller to provide the specified products or services and obligates the buyer to pay for them

• A document that clarifies responsibilities and sharpens focus on key requirements – deliverables, quality, timeframes etc

• A document that must be detailed and and accurately as they are used as the final position (you get out of them what you put to them)

• It is rarely used or relied on and seen as a last point of call.

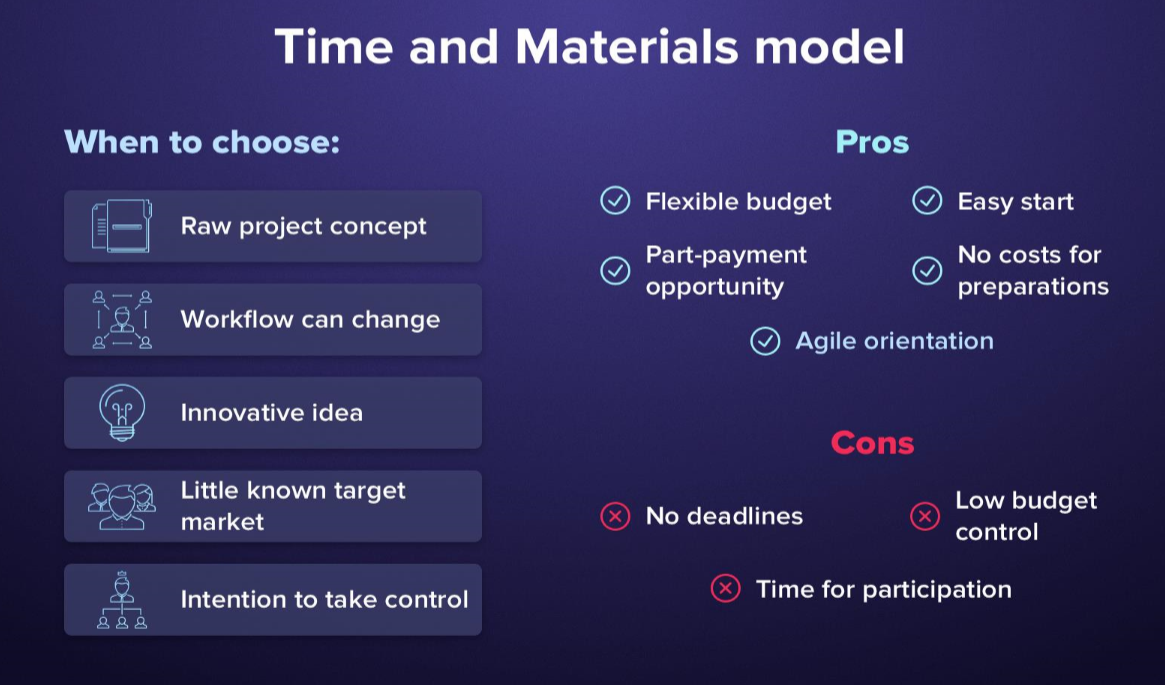
Different types of Contracts are used in different situations with all having pros and cons:

• **Fixed Price contracts**: involve a fixed total price for a well defined product or service.

• **Time & Material contracts**: involve payment to the seller for actual time spent and any materials used in providing the service.







Contracts should include specific clauses that take into account issues that are unique to the project – Quality, Time, Location etc Key contractual conditions should include • Intellectual Property Ownership and Indemnities

• Milestones and Deliverables

• Quality Criteria / Performance and Acceptance testing

• Variations / Change request process

• Non-Performance / Termination - Convenience, Breach etc

• Disengagement & Transition

• Liquidated Damages

• Fees and Penalties

• Warranties

**T9:**

**Australian Computer Society (ACS) Code of Professional Conduct**

**IEEE: Software Engineering Code of Ethics, Professional Practice(P3)**

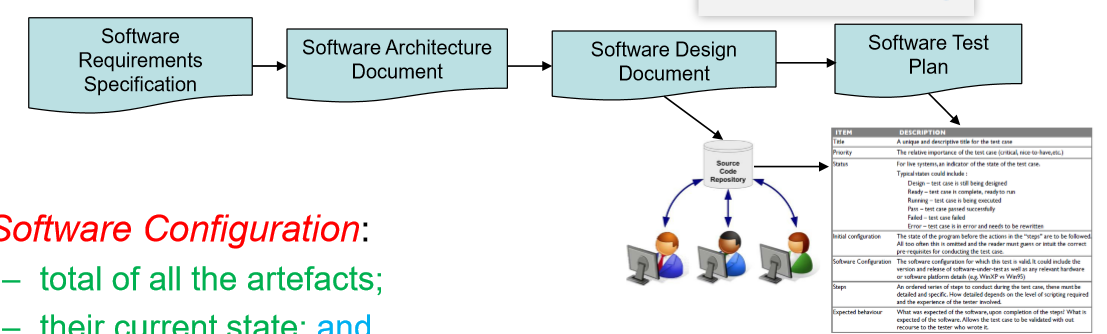
The quality metrics are based on a Service Level Agreement (SLA) contract

**Module 19:**

**Configuration Management**

Software Configuration:

• 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.

Role of Configuration Management

• 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?

**CM Processes**

CM Aims:

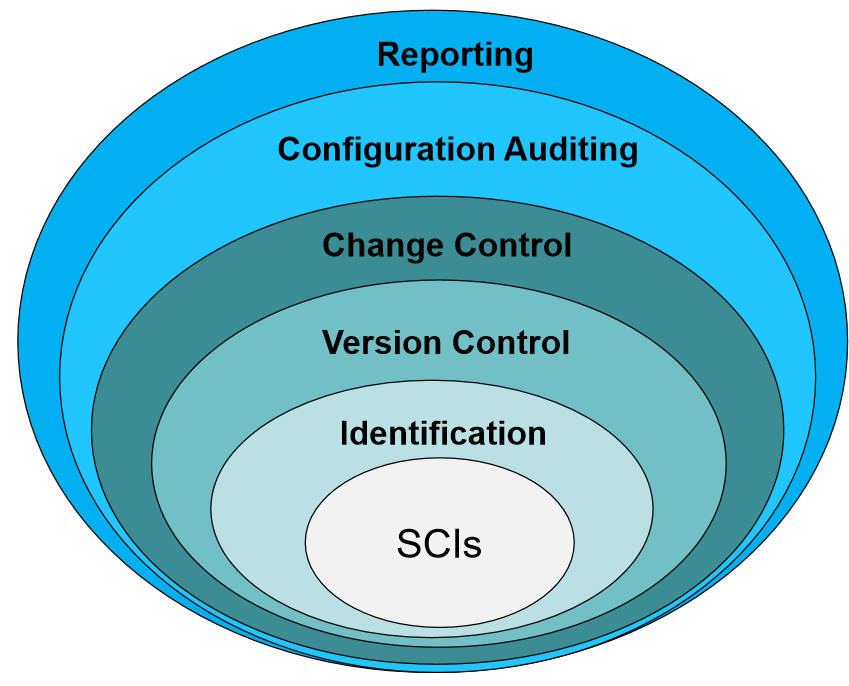
1. To identify all items that collectively will make up the configuration

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

CM Tasks



• 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

Identification

• 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

Version Control

• 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

**Change Control**

• 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



• 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 Auditing**

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

**Status Reporting**

• 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

**T10:**

Git:

Git Advantages:

- Distributed (everyone has their own code repository local to them!)

- Open Source (everyone likes open source code 😀😀)

- Bomb proof

