Projects 3
Challenges, risks and constraint 4
Development methodologies 5
Formal Models - waterfall, Incremental, V-Model 5
Agile - Kanban, Scrum, Extreme Programing 7
Agile Framework - Scrum 8
Risk Management 11
Risk identification techniques 12
Risk analysis and assessment 13
Respond to Risks 14
Project Management Plan 15
Stakeholders 16
Communication 17
Project scheduling 19
Gantt Chart 21
PERT Chart 21
EVA (earned value analysis) 22
Planning in Agile Development 23
Software cost estimation 24
Software Size Estimation - Source Lines of Code (SLOC) 25
Software Size Estimation - Function Points (FP) 25
Agile Effort Estimation 29
Agile Estimation Techniques 29
Motivation 31
Project Management & Leadership 31
Teams 32
Quality Management 34
Project Quality 34
Quality Assurance 34
Quality Planning 36
Quality Control and Monitoring 36
Outsourcing 38
Procurement 39
Sourcing Procurements 39

Managing procurements 41
Contracts 41
Ethics 42
Configuration Management 42
Software configuration: 42
CM Aims: 43
Identification 43
Version control 43
Change control 44
Configuration auditing 44
Status Reporting 44

Projects

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

Key characteristics of project:

- Introduce change to the organisation
- TEMPORARY, it has a defined beginning and end
- CROSS-FUNCTIONAL, cuts across organizational boundaries
- Deals with the UNKNOWN
- UNIQUE
- They all vary in SIZE: time, cost, scope

Why do organisations use Projects

- Provides strategic alignment of key activities and visibility at the appropriate levels
- Mechanism to prioritise activities (Benefits, Regulatory, HW Refresh)
- Allows organisations to deliver change in a structured and formal manner outside of Business As Usual
- 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 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.

Project manager key skills

- 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

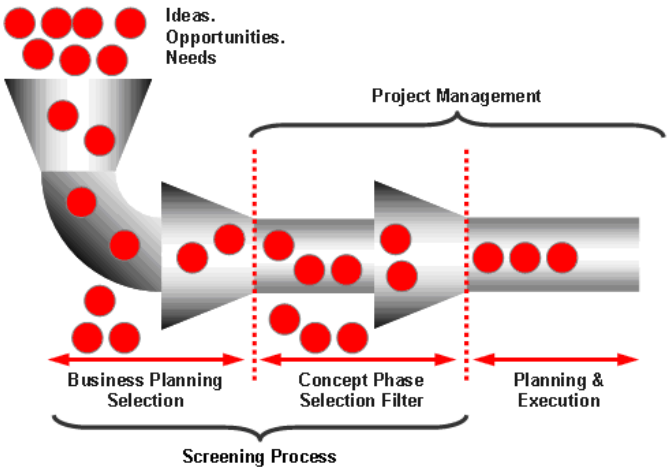
Project manager key activities

- **Planning**
  - Define and clarify project scope
  - Develop the project management plan and project schedule
  - Develop policies and procedures to support the achievement of the project objectives
- **Leading**
  - Setting team direction
  - Owning & coordinating activities across different organisational functions
  - Motivating team members
  - Assigning work
- **Organising**
  - Determine the project team structure
  - Identify roles and responsibilities
  - Identify services to be provided by external companies
  - Staff all project positions and on-going management

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

Success of software: hard to define, often over-time estimate or over budget.
Drivers to success: 60% of them are not technical issues and difficult to change.

How organisations select the best / right projects (Project Screening)



Before stating a project, think if the project really make sense, do you have enough confident that it is feasible and it could be successful

Business Case
The purpose of the **Business Case** is to establish mechanisms to judge whether the project is (and remains) desirable, viable and achievable as a means to support decision making in its initial and continued investment.
- Demonstrates how the project adds value to the organisation
- It is a living document throughout the project that should be reviewed and signed off at key stages

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

Project Charter
Contains: project Name, target date, project description, costs (resources, equipment and budget), gains (cost savings, time savings and revenue gain), project team, key milestones

Challenges, risks and constraint

Challenge are characteristic is known to exist. E.g. require fitness to travel by bike in cycle from Melbourne and Sydney
Constraints: could be cost, scope and time, there could be more e.g. communication due to covid
Risks: something may happen and leads to project failure, e.g. the algorithms required for project maybe too hard for developers

Development methodologies

Defined Process Control
A process with a well-defined set of steps. Given the same inputs, a defined process should produce the same output every time.
Great when in an environment with relatively low volatility that can be easily predicted; given the same inputs, a defined process should produce the same output every time based on its repeatability and predictability nature.

Empirical Process Control
In empirical process control, you expect the unexpected.
Empirical process control has the following characteristics:
- Learn as we progress
- Expect and embrace change
- Inspect and adapt using short development cycles
- Estimates are indicative only and may not be accurate

The **systems development life cycle (SDLC)**, a ***process*** for planning, creating, testing and deploying an information system.

Structured Project Management Methodologies e.g. PRINCE 2 etc
- Widely used and accepted - Consulting, Private and Government
- Process orientated approach
- Divides projects into multiple stages
- Detailed and thorough
- Must have a clear need, a target customer, realistic benefits, and a thorough cost analysis
- **Pros**: Extensive documentation is helpful with corporate planning & tracking
- **Cons**: Difficult and untimely to adapt changes and apply these to all documentation

When to choose a hybrid model?
- Client has a prescriptive model established

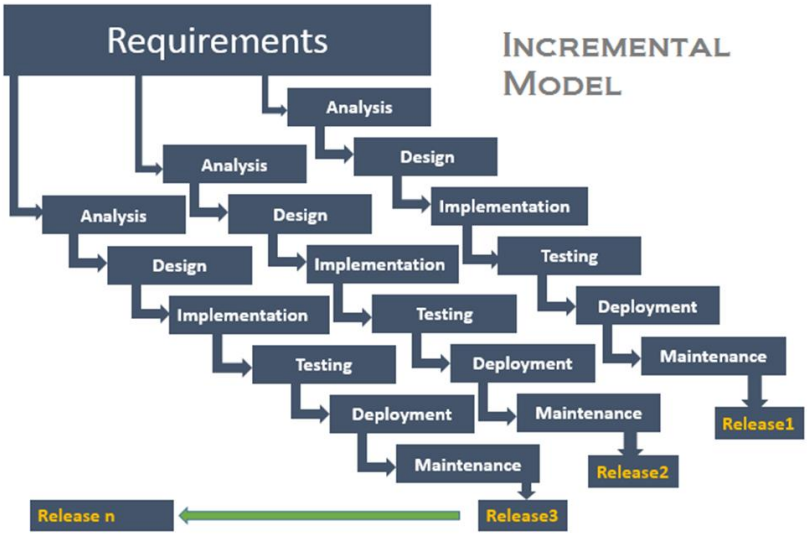
Formal Models – waterfall, Incremental, V–Model

Waterfall: Requirements -> Design -> Implementation -> Testing -> Maintenance
Always goes to next phase, and never go back
- 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
- **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

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.

What are the Advantages?
- Smaller, easier modules
- Initial modules released earlier
- Client feedback known earlier
- Change has less impact
- Requirements stable and precise

What are the Disadvantages?
- Management complexity
- Increased cost
- Partition skill
- Integration risk
- Rigid within each partition



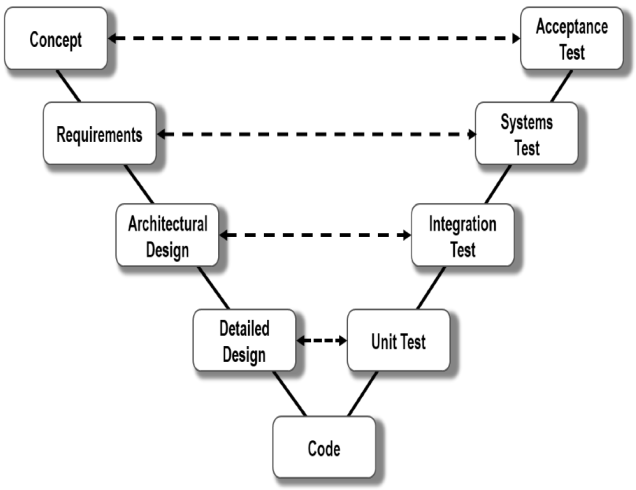
Advantages – compared to standard waterfall
- Each release delivers an operational product
- Less costly to change the scope/requirements
- Customers can respond to each build
- Initial product delivery is faster
- Customers get important functionality early
- Easier to test and debug during smaller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

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

V-model

What are the **Advantages**?
- Simple and easy management
- Rigid and sequential deliverable
- Documentation produced
- Requirements stable and precise

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



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

Agile – Kanban, Scrum, Extreme Programing

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

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

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

Agile Framework - Manifesto

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

Agile Framework - *12 Key Principles*

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes 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.

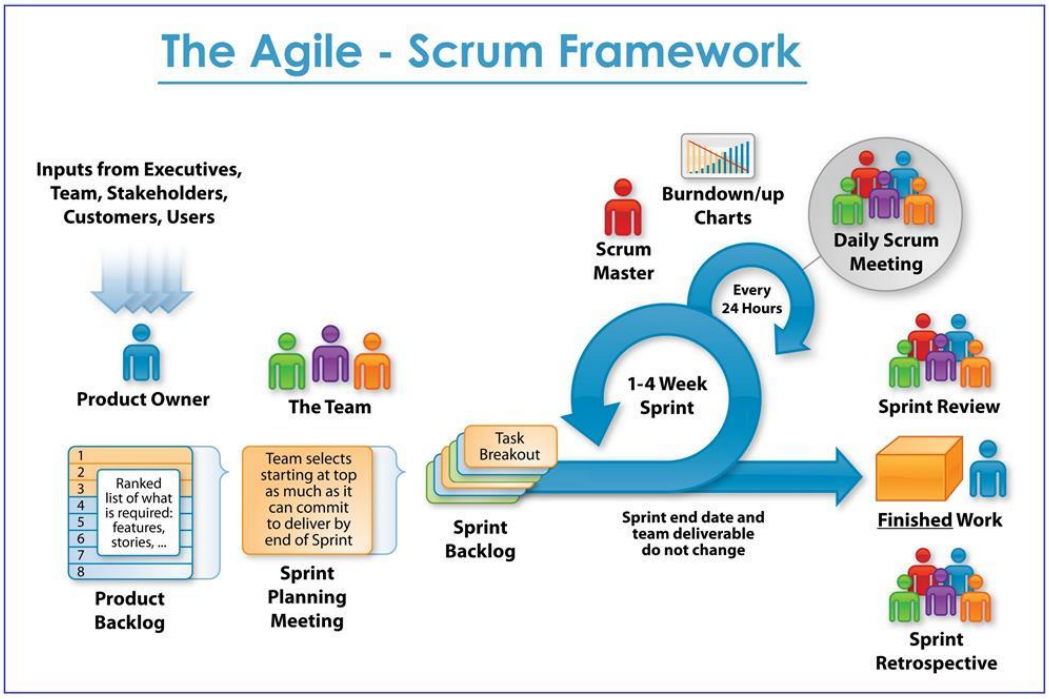
Agile Framework - Kanban
Visual progress gives transparency/accountability for self-organizing teams often referred to as ***SWIMLANE*** boards
Signboard / Billboard: Work items are visualised to provide participants a view of progress and process, from start to finish usually via a Kanban board. (ToDo, Doing, Done)

Agile Framework – Scrum

- 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)

Scrum Framework - *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 Roles – *Scrum Master*

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

Scrum Roles – *The Team*

- Cross-functional: Programmers, testers, user experience designers, business representatives etc.
- 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

Scrum Ceremonies / Meetings - Sprint Planning Meeting

1st Half of the meeting

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

2nd Half of the meeting

- Team figures out how the work will get done
- Break down each (feature) user story/large user story in the sprint backlog to capture all the work required (story points)
- The user stories form the basis of the sprint plan that is used to track, cost and manage progress

Scrum Ceremonies / Meetings - Sprint Reviews - Showcase

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

Scrum Ceremonies / Meetings - Daily Stand-up

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

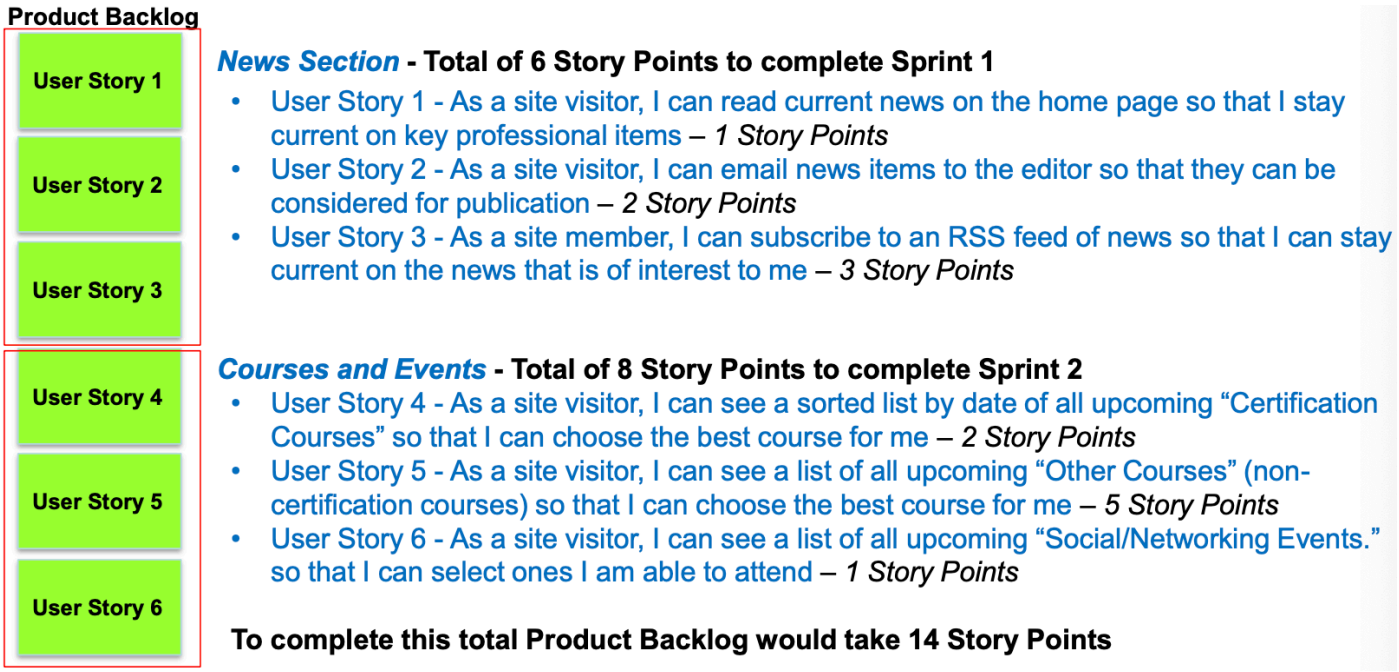
Scrum Ceremonies / Meetings - Sprint Retrospective

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

Scrum Artefacts – 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

Example



Scrum Artefacts – Product Backlog - User Stories

- 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 >
-

Scrum Artefacts – Product Backlog - Story Points

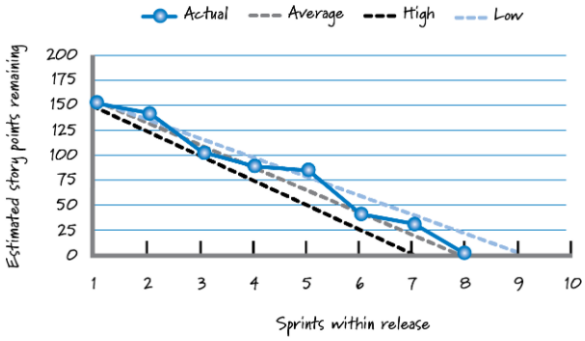
- Story points are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a product backlog item or any other piece of work
- Story points help estimate how much work can be done in a sprint

Scrum Artefacts – Sprint Backlog / User Story

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

Scrum Artefacts – Burn Down Chart

- A burndown 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.
- If it goes up: requirement change or re-assigned user story points



When to choose Agile Models?

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

Risk Management

A risk is a: Possible future event that has negative results
Risk is a result of uncertainty but not every uncertainty is a risk.

- Projects have many possible risks, that could have significant impacts on the outcomes:
  – Business risks, Project risks, Product risks
- The goal of project risk management is: **minimising the impact of potential negative risks while maximising the impact of potential positive risks**

Risk management loop: **Identify -> Analyze -> Action -> Monitor -> Control -> Identify**

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 organization’s and the sponsor’s approaches to risk management
- The level of detail will vary with the needs of the project

The Risk Management Plan Template

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)
- **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
- **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
- 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 techniques

Pondering

- This simply involves an individual taking the "pencil and paper" approach of risk identification, which involves sitting and thinking about the possible risks that could occur in the project

Interviews/questionnaires

- Interviewing project stakeholders, or asking them to fill out questionnaires, to harness their knowledge of a domain
- It is unlikely that a risk manager in a software project will have sufficient knowledge of the methods and tools to be employed to provide a comprehensive view of the risks, so input from stakeholder and domain experts is essential

Brainstorming

- The team can use a risk framework or the Work Breakdown Structure (WBS) to identify threats and opportunities
- The key is to encourage contributions from everybody, then discuss and evaluate

Checklists

- This involves the use of standard checklists of possible risk drivers that are collated from experience
- These checklists are used as triggers for experts to think about the possible types of risks in that area

Delphi Technique

- A group of experts are asked to identify risks and their impact
- The responses are them made available to each other anonymously
- The experts are then asked to update their response based on the responses of others
- repeated until consensus is reached

SWOT Analysis (Case study)

- Strengths, Weaknesses, Opportunities and Threats
- This technique allows finding strengths and weaknesses as well

- *Strengths*: characteristics of the business or project that give it an advantage over others
- *Weaknesses*: characteristics that place the business or project at a disadvantage relative to others
- *Opportunities*: elements in the environment that the business or project could exploit to its advantage
- *Threats*: elements in the environment that could cause trouble for the business or project

Risk analysis and assessment

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 - Qualitative

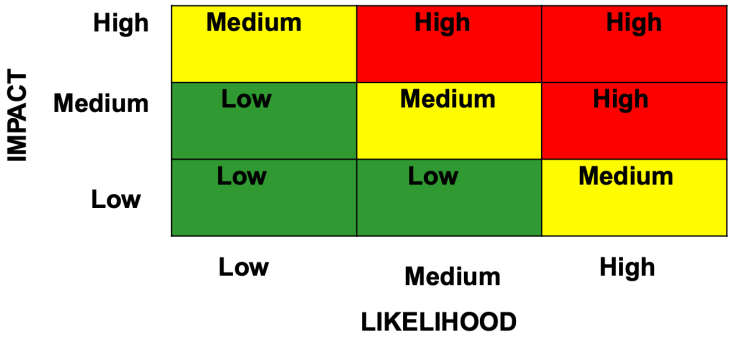
The important steps of risk analysis are:

1. Estimating the risk probability (P)
   - this is an estimation of the probability that the risk will occur
2. Estimating the risk impact (I)
   - the impact that the risk will have on the project
   - Usually measured in a scale of 1 – 5 (or 10): (1)no impact; (2) minimal impact; (3) moderate impact; (4) severe impact; and (5) catastrophic impact
   - Impact can be expressed as a monitory value
3. Compute risk exposure (or P * I Score) Risk exposure= $P * I$
4. Identifying the root cause
   - It is important that one identifies the root causes of all risks
   - If this root cause can be identified, then all of these risks can be controlled by addressing the root cause

Example

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

Risk Assessment - ***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

Quantitative approaches include mathematical and statistical techniques
They are based on modeling a particular risk situation - probability distributions of risks are the main consideration
• Common Techniques: Decision Tree Analysis, Simulation, Sensitivity Analysis

Respond to Risks

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 project category 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

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

Risk ID	Trigger	Owner	Response	Resources Required

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)

Agile Risk Management

Identify – Capture risks in Product Backlog
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 & Planning
Scrum master manages risk daily by removing impediments - just in time
Product owner- Business risk managed by product owner; Add risk items; Prioritisation of Product Backlog

Project Management Plan

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.

Project Charter V Project Management Plan

A Project Charter is a summary project proposal to secure approval for the project goals and terms (useful as part of Business Case).
A PMP is an approved document showing how to achieve the approved project goals / benefits and provides the details on how to execute and manage the project (used as part of mobilisation and on-going management of the project).

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
    - Schedule, Risk Management, Cost Estimation, QualityAssurance, Configuration Management (Change Management)

Stakeholders

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

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 (high or low power?)
- Suggestions and Strategies for Managing Relationships with each Stakeholder

Communication

Communication Challenges
Individual challenges: Semantics [meaning], Perception [interpretation], Communication Channel, Feedback, Anxiety, Culture
Team challenges: Status, Silos, Information Overload, Lack of Communication, Protocol [rules]

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

Communication skills
- 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

Communication Importance
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 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

Virtual Teams & Communication

- Communication is less rich and less frequent than face-to-face interaction
  - Less visual and behavioral cues
  - Less or no informal interactions
- Those less inclined to speak in groups, may feel more comfortable
- Less importance on interpersonal skills and physical appearance may benefit certain members of the tea
- Still need to be mindful of unconscious bias (virtual unconscious bias)

Create a Communication charter.

- Discipline about how the team should communicate
- Norms of behavior 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

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

Items to remember

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

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

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

Project scheduling

- Is used and maintained throughout the project to monitor and track project progress
- Is a living document
- **What does the project schedule contain**?
  - Duration and dependencies for each task
  - People and physical resources required by each task
  - Milestones and deliverables
  - Project Timeline

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

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

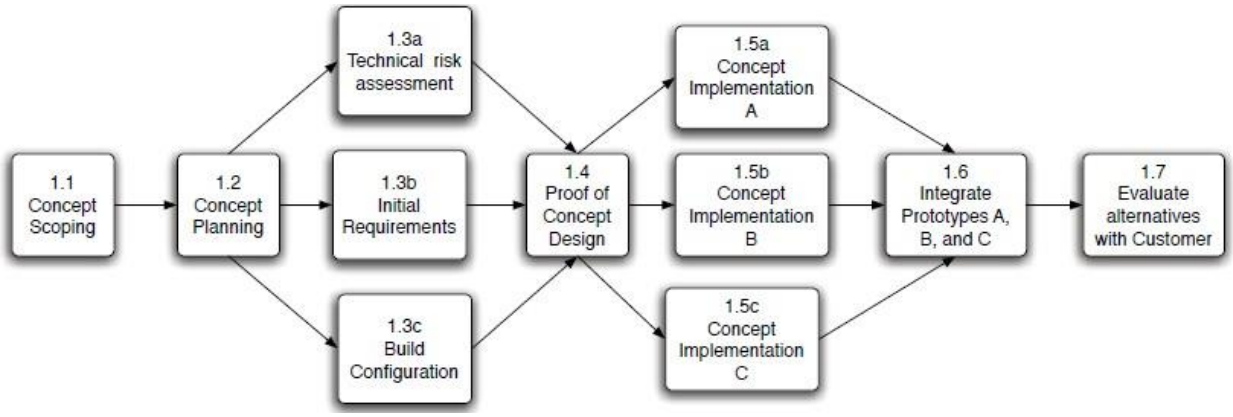
Developing Project Schedule - Steps

1. Breakdown the task into manageable units – result in a 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

Identify Task Dependencies - Step 2

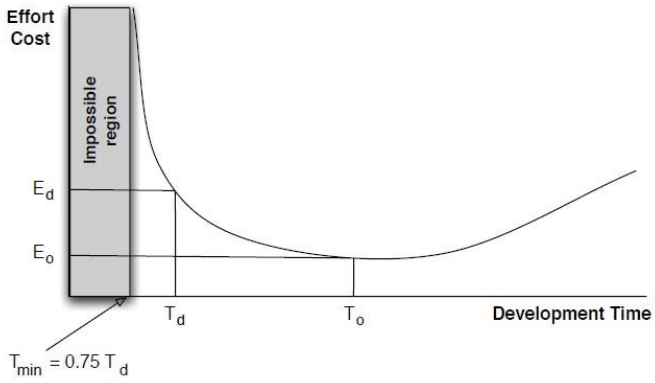
- Tasks can be:
  - Unconstrained: the task can start at any time
  - Constrained: depends on another task
    - If task B depends on task A (A -> B), B is a Successor task (S), A is a Predecessor task (P)
- Dependencies are caused by:
  - a task needing a work product of another task
  - a task needing resources used by another task

Task Network



Effort-time Estimation

- 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
optimistic time - O
pessimistic time - P
most likely time - M
expected time - T_E
$T_E = (O + 4M + P) / 6$
(Which is average of 6 people's idea)

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

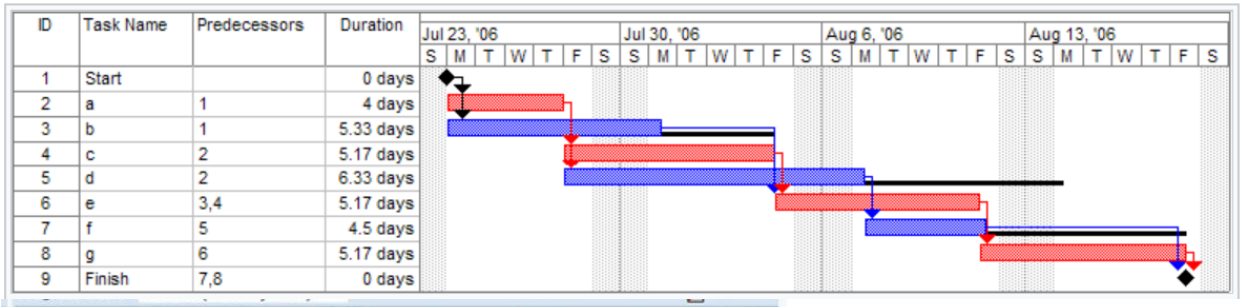
Project Scheduling - Definition
Activity (Task): Is part of a project that requires resources and time
Milestone: Is the completion of an activity that provides evidence of a deliverable completion or end of a phase – is an event that takes zero time
Free float (free slack): Is the amount of time that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

Milestones vs Deliverables
- 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

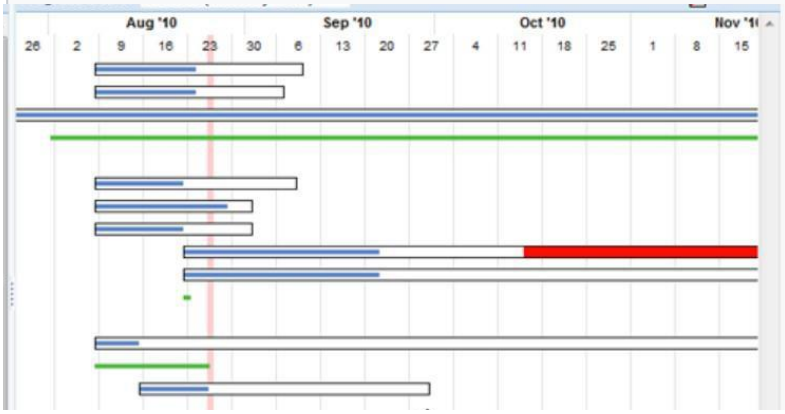
Gantt Chart

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

Linked Gantt charts: contain lines indicating the dependencies between tasks



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

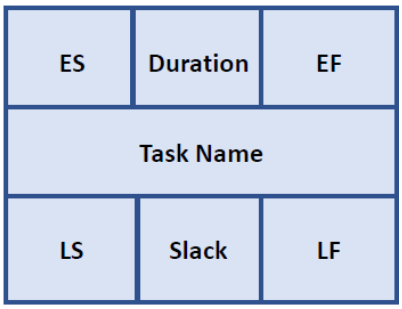


Progress Gantt charts
- tasks are shaded in proportion to the degree of their completion
- used for progress tracking – gives a visual representation of the progress

PERT Chart

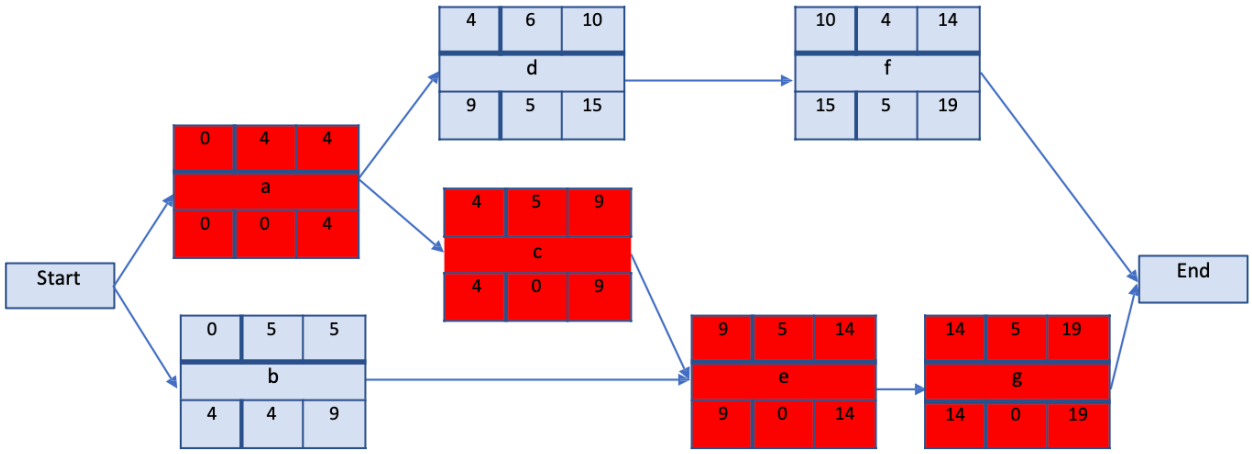
Program Evaluation and Review Technique chart

- 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**):
    - works dependent on nothing ES = 0,
    - if it dependent on 1 work, pick the work's EF as its ES,
    - if it dependent on 2+ works, pick the one with largest EF
  - Latest start time (**LS**):
    - = LF - Duration
  - Earliest finish time (**EF**):
    - = ES + Duration
  - Latest finish time (**LF**):
    - if it's the one points to the end, LF = largest EF
    - If one work dependent on this work, pick its LS
    - if 2+ works dependent on this, pick the one with smallest LS
  - **Slack** time
    - = LF - EF



Order to complete: ES, Duration and EF of all from start to end, then LF, LS from end to start, then Slack

Critical Path Methods



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

- 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

EVA (earned value analysis)

A formal method to track and control project progress

- Planned Value (**PV**)
  – that portion of the approved cost estimate planned to be spent on the given activity during a given period, = Estimate cost * actual time cost / total estimate time
- The Earned Value (**EV**)
  – the value of the work actually completed = Estimate cost * %work done
- Actual Cost (**AC**)
  – the total of the costs incurred in accomplishing work on the activity in a given period

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

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

EV = $100,000 * 20 / 100 = $20,000 AC = $35,000

- **Schedule Variance Analysis: expressed in $**
  SV = EV - PV = 20,000 – 25,000 = (5000)
- **Schedule Performance Index: expressed as a fraction**
  SPI = EV / PV = 20,000/25,000 = 0.8
- **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

Planning in Agile Development

Planning in Scrum

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

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
• **Agile Planning**
- 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 (used when date is more important)

- Determine the number of sprints N = total duration / length of sprint
- Groom the product backlog by estimating and prioritizing stories
- Measure team velocity range V_{min}, V_{max}
- Compute the minimum and maximum story points based on velocity
  $SP_{min} = V_{min} * N$
  $SP_{max} = V_{max} * N$
- Draw lines through the product backlog to show the above

Fixed-Scope Release Planning (used when scope is more important)

- Groom the product backlog by creating, estimating and prioritizing and identify the must-have stories
- Determine the total number of must-have story points (SP_{Total})
- Measure team velocity range: V_{min}, V_{max}
- Compute minimum and maximum number of sprints (round to integer)
  $S_{min} = SP_{Total} / V_{max}$
  $S_{max} = SP_{Total} / V_{min}$
- Show on Burndown Chart

Software cost estimation

Estimation of how much money, effort, resources, and time will take to build a specific software based system or product

Importance: 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

• 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

Some 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 (if putting too much effort, accuracy drops)
- Know that you cannot eliminate uncertainty from estimates but small efforts are rewarded with big gains

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: $e = (p + 4m + o)/6$
- *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 × $Size^B$ × 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 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

Software Size Estimation – Source Lines of Code (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
- For the clients don't have programing background, it's difficult for them to understand

Software Size Estimation – Function Points (FP)

- Is used to express the amount of functionality in a software system, as seen by the user
- Typically used to:
  - Estimate the cost and effort required to design, code and test a software system
  - Predict the number of errors
  - Predict the number of components
  - Measure productivity
- Function points are computed from the Software Requirements Specification (SRS)

Advantages of Function Points

- Measures the size of the solution instead of the size of the problem
- Requirements are the only thing needed for function points count
- Can be estimated early in analysis and design
- Is independent of technology
- Is independent of programming languages

Disadvantages of Function Points

- A well defined requirements specification is necessary
- Gaining proficiency is not easy, the learning curve is quite long
- Could be quite time-consuming thus could be costly

FP Computation Steps

1. Categorize requirements
2. Estimate a complexity value for each category or function
3. Compute *count total* from Complexity
4. Estimate value adjustment factors
5. Compute total function point count

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 (quality requirements)

1. Categorize Requirement

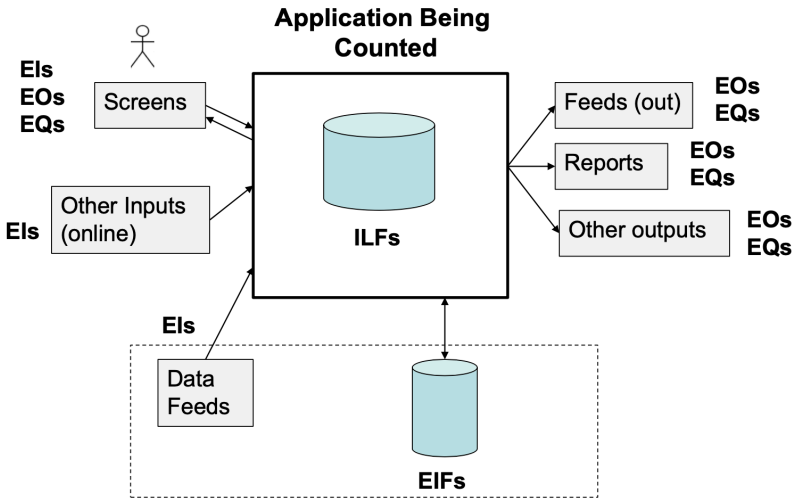
Function (Requirement)

Data functions: Concerned with maintenance of the data for the application

- 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 - data hosted on a third-party servers

Transaction Functions: Concerned with information being passes to and from the system

- 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, 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



R.1	Read the previous day's trading information (high price, low price, opening price, closing price) from a third-party-server.	EI
R.2	Save a complete "price history" in a database.	ILF
R.3	Based on the trends in prices for commodities, decide which commodities to bid for, and which to try to sell.	
R.4	Send information to an external e-trading account which places the bid/ask for each commodity.	EIF
R.5	When the bid/ask is either accepted or expires, record the result in a "transaction history" database.	ILF
R.6	At the end of the day, produce a report that summaries the transactions of the day along with the following information: profit/loss for the day; number of trades of each commodity; average market price of all commodities; account summary.	EO
R.7	At anytime the user can request a transaction history for a period which gives: transaction IDs; commodity type; bid/ask; price.	EQ

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

Relationship between DETs, RETs, FTRs, and the function categories

Function	DETs	RETs	FTRs
Internal Logical Files (ILFs)	x	x	
External Interface Files (EIFs)	x	x	
External Inputs (EIs)	x		x
External Outputs (EOs)	x		x
External Inquiries (EQs)	x		x

Complexity table for Data Functions

	DETs		
RETs	1-19	20-50	51+
1	Simple	Simple	Average
2-5	Simple	Average	Complex
6+	Average	Complex	Complex

Complexity table for Transaction Functions

	DETs		
FTRs	1-5	6-19	20+
1	Simple	Simple	Average
2-3	Simple	Average	Complex
4+	Average	Complex	Complex

Automated Trading System: Function Categories

			DETs	RETs	FTRs	Complexity
R.1	EI	high price, low price, opening price, closing price, date	5		1	Simple
R.2	ILF	commodity name, high price, low price, opening price, closing price, date	6	1		Simple
R.3						

R.4	EIF	Commodity, bid/asking price, buy/sell	3	1		Simple
R.5	ILF	...	4	1		Simple
R.6	EO	...	5		3	Simple
R.7	EQ	...	4		2	Simple

3. Compute *count total* from Complexity

• Using count and complexity estimates compute total count

Information Domain Value	Count		Weighting Factor Simple	Average	Complex		
Internal Logical Files (ILFs)	2	×	7	~~10~~	~~15~~	=	14
External Interface Files (EIFs)	1	×	5	~~7~~	~~10~~	=	5
External Inputs (EIs)	1	×	3	4	~~6~~	=	3
External Outputs (EOs)	1	×	4	~~5~~	~~7~~	=	4
External Inquiries (EQs)	1	×	3	4	~~6~~	=	3
Count total							29

4. Compute value adjustment factors

• each of the characteristics is ranked on a scale of 0-5; 0 not important and 5 critical
  - Data Communications
  - Distributed Data Processing
  - Performance
  - Heavily Used Configuration
  - Transaction Rate
  - Online Data Entry
  - End-User Efficiency
  - Online Update
  - Complex Processing
  - Reusability
  - Installation Ease
  - Operational Ease
  - Multiple Sites
  - Facilitate Change

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

$FP = \text{count total} \times (0.65 + 0.01 \times \Sigma^{14}_{i=1} F_i)$

F_i - VAF corresponding to the i-th VAF question

e.g. FP = 29 x (0.65 + 0.01*38) = 29 x1.03 = 29.87

Agile 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)

- (Sprint) User Story
  - Detailed technical level; A developer's perspective; A conversation placeholder
- Feature User Story
  - Product capabilities; Business level detail; Product Owner perspective
- Epic User Story
  - Lacks detail; New business services; A product

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

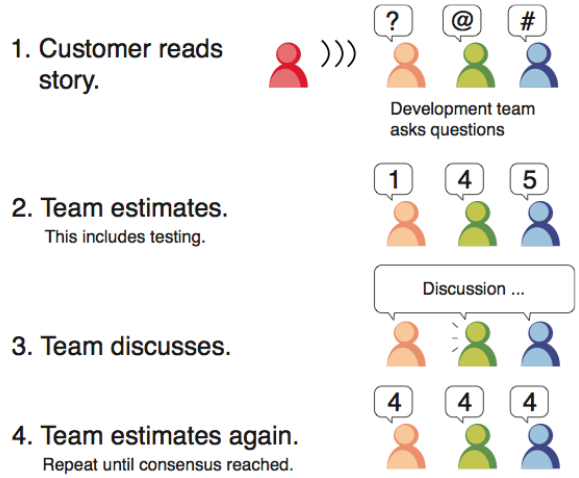
1. Develop user stories for the system.
2. Estimate the number of story points for each story, basing the estimate on the number of story points from previous stories, using a chosen technique (discussed later).
3. Use the team's velocity from previous experience to estimate the delivery time of the 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.

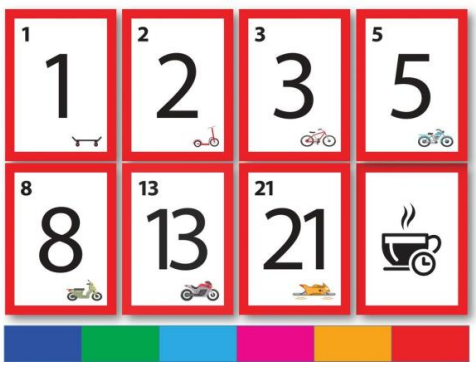
Agile Estimation Guidelines

- **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

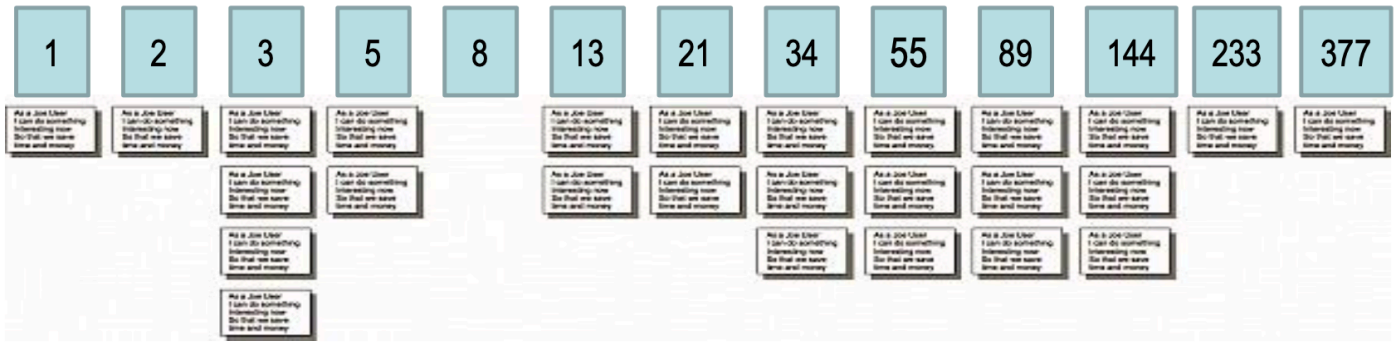
Agile Estimation Techniques

Planning Poker





BUCKET SYSTEM



- The team sitting at a table picks a user story card randomly and places it in bucket 8
- The next few cards are randomly picked one at a time, discussed agreed on, and placed in a bucket relative to the previous ones
- Then each person is allocated a set of cards and they are placed in a appropriate bucket, based on individual judgement (Divide and conquer)
- Finally the team reviews the placements and reach agreement

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

V = SP / Ti
V - velocity
SP - number of story points completed
Ti - time period over which they were completed
Velocity: Using historical data; Using data from previous iterations

Estimated Delivery Time

$T = (\Sigma^{n}_{i=1} SP_i) / v$
T - estimate delivery time
V - velocity
SP_i - number of story points in the i-th user story
n - total number of user stories

Final comments

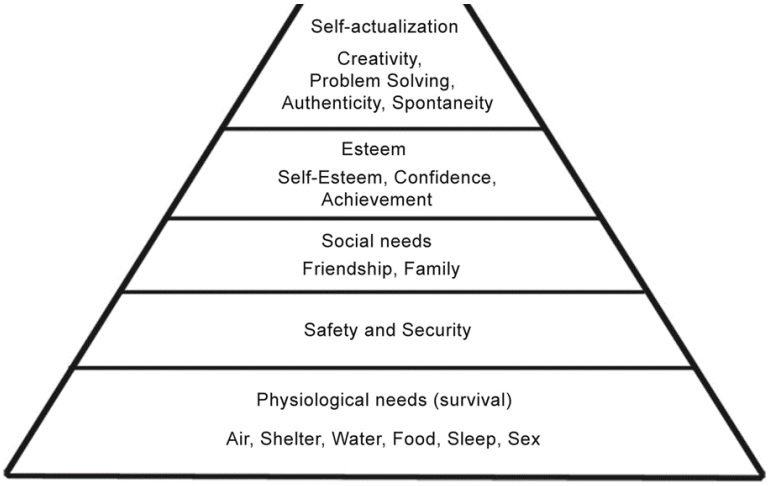
- Allow enough time to do a proper project estimate - rushed estimates are inaccurate, high-risk estimates
- There is diminishing return on time spent estimating
- Know that you cannot eliminate uncertainty from estimates but small efforts are rewarded with big gains

Motivation

Motivation can inspire, encourage, and stimulate individuals and project teams to achieve great accomplishments. Motivation can also create an environment that fosters teamwork and collective initiatives to reach common goals or objectives.
It is the driving force within individuals that propels them into action

Maslow Hierarchy of 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

- Asked people to describe situations when they felt really good and really bad about their jobs
- Remedying the causes of dissatisfaction will not create satisfaction

Project Management & Leadership

- 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
- 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

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

Teams

- An individual is a person with a unique set of skills
- A Team is two or more individuals consciously working together to achieve a common objective

Why teams

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

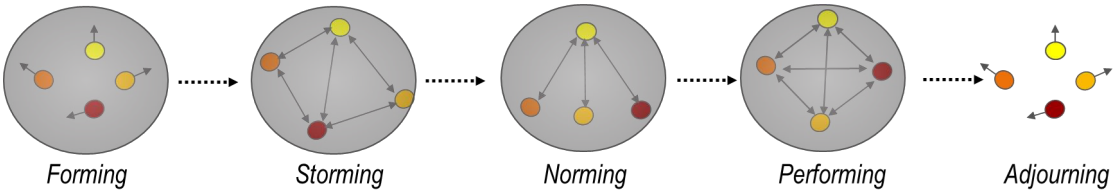
Benefits / value of teams

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 socialize, 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.

How Teams Form & Perform

Tuckman's Team Development Model

1. Forming, 2. Storming, 3. Norming, 4. Performing



Is the Team Effective or Not

Positive signs

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

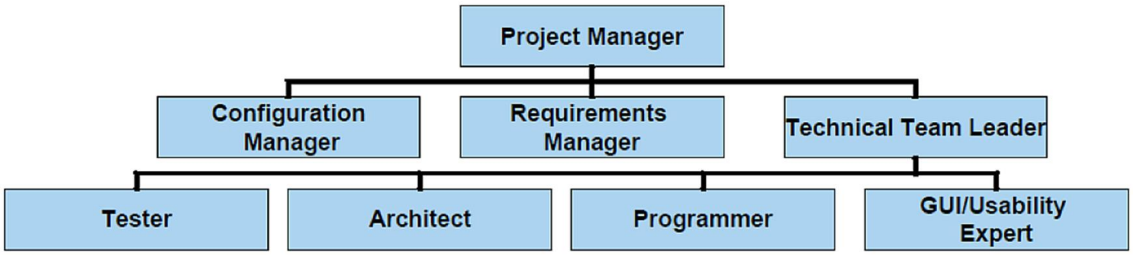
Not so Positive signs

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

Teams Structures

Controlled Centralised

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



Scrum Team

- Used in Agile
- Structure is a lot more flat (dev team, scrum master, project owner)

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 take 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

Quality Management

Project Quality

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

Quality Management Process
Quality assurance: The establishment of a framework of organizational procedures and standards that lead to high-quality software. Including verification and validation
Quality planning: The selection of appropriate procedures and standards from the framework, adopted for the specific project
Quality control: Ensuring that the software development team has followed the project quality procedures and standards

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).
Tests
– Code —> Unit test
– Design —> Integration
– Requirements —> Validation test
– System engineering —> System test

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
- Product standards:
  • These apply to the product being developed
- Process standards:
  • These standards define the processes that should be followed during software development

Product vs Process Standards

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

Documentation standards

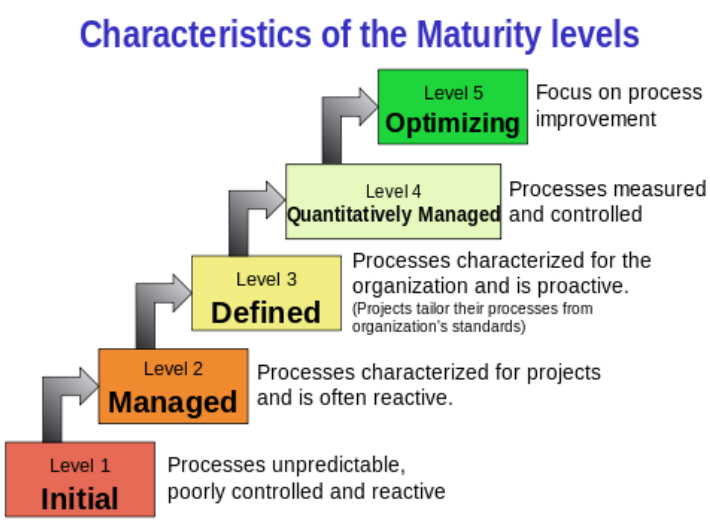
- Documents are the tangible manifestation of the software
- Documentation process standards
  – How documents should be developed, validated and maintained
- Document standards
  – Concerned with document identification, structure, presentation, changes highlighting, etc.
- Document interchange standards
  – How documents are stored and interchanged between different documentation systems

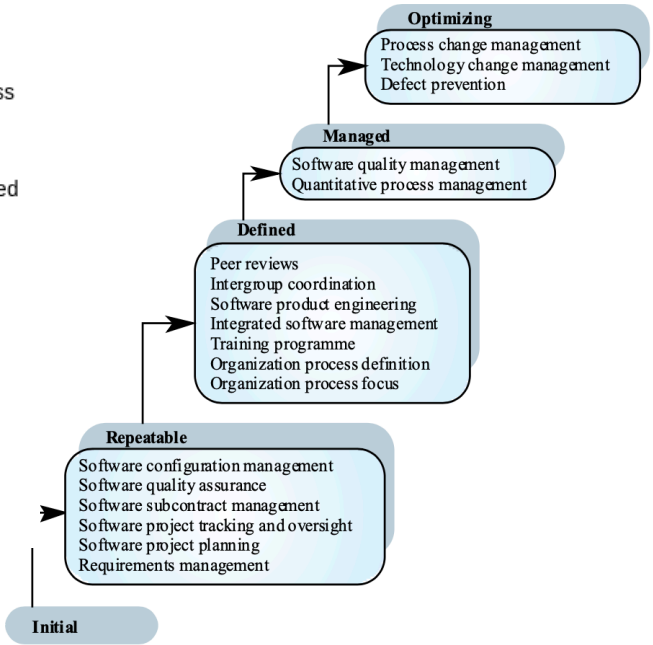
Software Standards and Systems

- 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

Capability Maturity Model

- 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)

SQP - Template

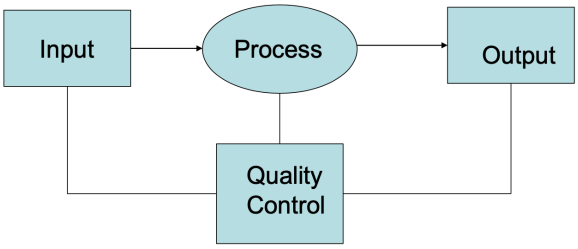
- Product Overview - A description of the product, intended market, and quality expectations
- Product Plan - The critical release dates and responsibilities – could point to the schedule
- Quality Goals - The quality goals and plans for the product, including identification and justification of critical 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

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

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



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

- **Advantages of technical reviews**:
  – Can perform on all artefacts
  – Earlier detection of problems in software artefacts leads to lower costs of resolution.
  – Review techniques can find several types of faults that testing failed to find.
  – Reviews find the actual faults in source code, it can be located.
  – Some programmers may correct faults in reviewing phase
- **Disadvantages of technical reviews:**
  – Could be time and resource consuming
  – Should be carefully planned and executed to get the desired outcomes
- **Types of technical reviews**
  – Informal Reviews
  – Formal Reviews
  – Walk throughs
  – Code inspections
  – Audits
- **Informal Reviews:**
  – A simple meeting with a colleague which aims to improve the quality of a document
  – No formal guidelines or procedures that are followed
  – 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**
  – 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
- May involve reviewing if the project meets the scope, schedule, budget and quality objectives

Outsourcing

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.

Various activities are better suited to the type of Outsourcing:
Architecture, Change Management, Project Management, BusinessAnalysis, Design, Software Development Testing, Operational (Application & Infrastructure) Support

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 local job markets
- Loss of Relationship building opportunity with key stakeholders

Procurement

The Procurement Management Process consists of 3 broad stages:

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.

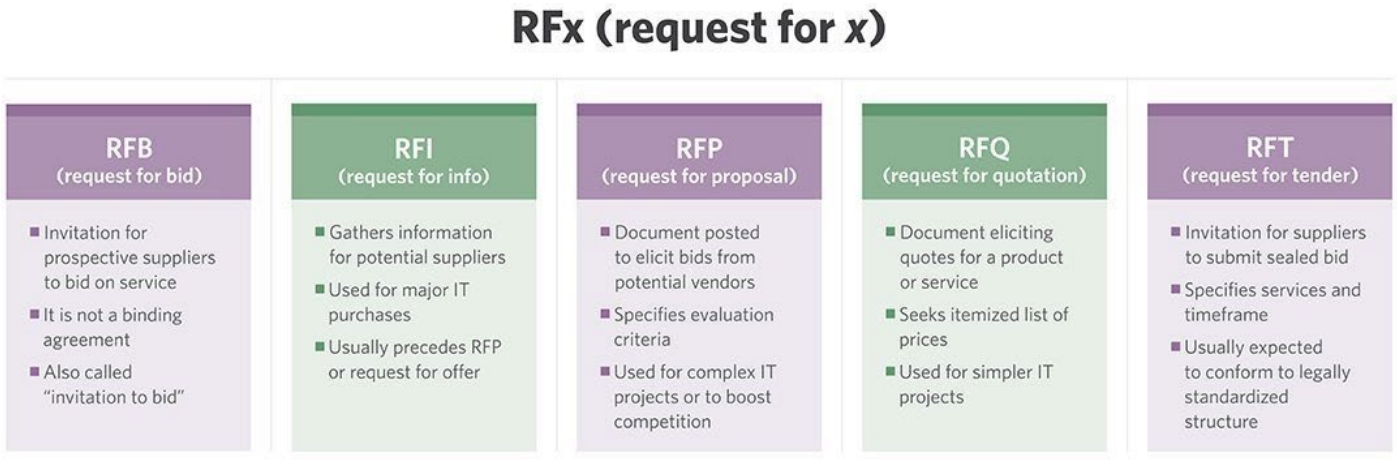
Source, the principal objective of this stage is to identify and engage suppliers who will provide the best value for money outcome, in a framework of probity and fair dealing. A key deliverable for this stage is to determine the appropriate sourcing method, with consideration given to alternatives other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

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



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

The RFx document will typically include:

Purpose of RFx; Organisation's Background; Basic Requirements; Hardware and Software Environments; Description of RFx Processes & Evaluation; Statement of Work and Scheduled Information.

Appendices: Current Systems Overview; Systems Requirements; Volume and Size data; Required Contents of Vendor's Response to RFx; Sample Contract.

SOW - *Statement of Work*

A description of the work required. Gives bidders an understanding of buyer's expectations:

- 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

Sourcing procurement process:

- 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.

Sample Evaluation Sheet:

		Proposal 1		Proposal 2		Proposal 3	
Criteria	Weight	Rating	Score	Rating	Score	Rating	Score
Technical Approach	25%						
Management Approach	25%						
Past Performance	20%						
Price	30%						
Total Score	100%						

To calculate the score multiply the weight of the criterion by the rating for the proposal

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

Contracts

- 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

***Fixed Price* contracts:**
involve a fixed total price for a well-defined product or service.
Suitable for waterfall
Risks: buyer's risk low, seller's risk high,
choose when:

- deadline is clear,
- specification is precise,
- short project duration,
- optional client's control,
- no change planed.

Pros:

- No overpayments;
- No distrust; No supervision;
- No turn-ups

Cons:

- Long time preparation;
- Minor control over the process;
- Lack of communication

Time & Material contracts: involve payment to the seller for actual time spent and any materials used in providing the service.
Suitable for agile
Risk: buyer's risk high, seller's risk low,
Choose when:

- Raw project concept
- Workflow can change
- Innovative idea
- Little known target market
- Intention to take control

Pros:

- Flexible budget
- Easy start
- Part-payment opportunity
- No cost for preparations
- Agile orientation

Cons:

- No deadlines
- Low budget control

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

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
- **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.

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.

Configuration Management

Software configuration:

- *Total of all artefacts*, such as use-case diagrams, class diagrams, test cases, source code, Software Requirements Specification, Software Architecture Document, Software Design Document, Software Test Plan
- *The artifacts current state*
- *The dependencies between artefacts*, such as, a code module may depend on design diagram

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
   – establishing processes;
   – setting up repositories; and
   – using other appropriate tools and techniques
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: one class in code/ one single document;
  - Aggregate: all classes in code/ all documents ;
  - Derived: the output after compiling

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

• 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: Same as a Version, with non-functional changes. 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
  e.g. V1.0, V2.0, V2.1…

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

Element	Impact on the Process
Initiate the Change	• Why is the change being made? • What information will be needed to evaluate the change? • How will the change be evaluated?
Evaluate the Change	• How will the change affect the configuration? • Which artefacts need to change and what are their dependencies? • What are the benefits of the change? • What are the costs of the change? • Do the benefits of the change outweigh the costs of the change? • Who will be impacted by the change?
Making the Change	• Who will put the change into effect? • How will the change be managed? • How will other people working on the project understand the change? • How will they be notified of the change? • How will people working on the project know when the change is completed?

• Baseline
– A baseline is an artefact that is *stable*
– It has been formally reviewed and agreed upon, that is now ready for future development
– It can only be changed through a *formal change management procedure*

Configuration auditing

– 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

– 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