Table of Contents

C1: Project Management Concepts

Project Management Disciplines

Disciplines	Description
Planning	Deciding what to be done (Project Initiation Document [PID] needs to be created)
Organising	Making arrangement
Staffing	Select personnel
Directing	Give instructions
Monitoring	Check on progress
Controlling	Remedy hold-ups 排除故障
Innovating	Coming up with new solutions

Characteristics of Projects

- **Non-routine** (e.g. FYP)
    - Projects are unique, not repetitive daily operations
    - Each project requires fresh thinking, tailored methods and problem-solving
- **Planning is required** (e.g. what needs to be done...)
    - Projects do not succeed by chance - they need structured planing
    - Planning covers scope, tasks, timelines, risks and resources
    - Without planning, projects drift, overshoot budgets or fail to meet objectives
- **Specific objectives to meet** (e.g. Develop a student attendance system to reduce the amount of paper consumption by 99%)
    - Every project has a clear goal
    - Objectives are measurable, so success can be evaluated
    - Objectives keep the team aligned and provide criteria for judging completion

- **Pre-determined time span** (e.g. FYP 9 months)
    - Projects are temporary - they have a start and an end
    - Time constraints force prioritization and efficiency
- **Work is carried out for someone** (e.g. clients)
    - Projects serve a client, sponsor or stakeholder
    - Even in academic projects, the "client" could be your supervisor and examiners
    - Deliverables must meet stakeholder expectations, not just the team's preferences
- **Involves several specialisms** (e.g. database specialists, UI specialist, Network specialist, etc)
    - Projects often require cross-disciplinary expertise:
        - Database design
        - User Interface (UI) design
        - Networking / security
    - Collaboration is essential; no single person usually has all the skills
- **Involves several phases** (e.g. analysis, design, code, test, implement, deployment)
    - Projects follow a life cycle: Analysis, design, coding, testing, implementation, deployment
    - Breaking work into phases make it manageable and ensures quality checks at each stage
- **Resource constraints are common** (e.g. Time, budget, staffs)
    - Projects always face limits: time (deadlines), budget (funding), staff (availability of skilled people)
    - Constraints force trade-offs - e.g. you may need to reduce features to meet deadlines

Difference between Software Project and Other Projects

	Software Project	**Other Types of Project**
Invisibility	Progress is not immediately visible (no physical form). All is within the code. *You cannot watch code being built like a bridge.*	Progress is immediately visible (have physical form)

Conformity	Have to conform to clients requirements *Software obeys people*	Have to conform to physical laws (e.g. gravity law) *Bridges obey physics*
Complexity	Have more complexity than other engineered artefacts More hidden, abstract complexity *Software has invisible spiders in the web*	Less complex than software project
Flexibility	Easier to change (edit code) *Changing code is Ctrl + Z*	Difficult to change once it is completed (rebuild needed) *Changing concrete is bulldozer*

Constraints in Projects

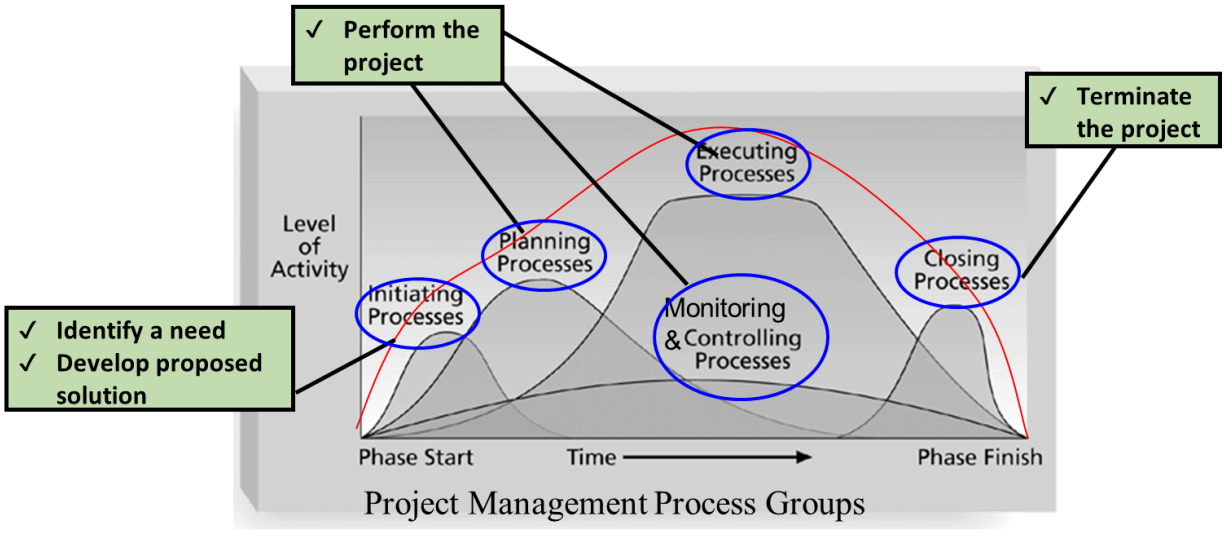
- Time
- Budget
- Quality

Challenges in Software Projects

Position	**Challenges**
Project Manager	• Poor estimates and plans • Lack of quality standards and measures • Lack of guidance about making organisational decisions • Lack of techniques to make progress visible • Poor role definition - who does what • Incorrect success criteria • Impossible deadlines • Scope changes

	• Resource constraints
Developers	• Inadequate specification of work • Management ignorance of IT • Lack of knowledge of application area • Lack of standards • Lack of up-to-date documentations • Preceding activities not completed on time • Narrow scope of technical expertise • Changing user requirement • Deadline pressure • Lack of skill • Lack of communication
Customers	• The software project is behind schedule (e.g. need to wait 1 more month then only can use the software) • The developed software no longer meets the needs of the organization • The cost of the project has exceeded its allotted budget

Project Life Cycle Processes



Phase	Description
Initiating Processes	• **Conduct a kick-off meeting** to establish customer's needs, goals, requirements and expected benefits • **1 or 2 more meetings to streamline the project requirements** • **Output: Project Charter** (formal project authorization which documents shared understanding of project scope, development and objectives with role and responsibility definition)
Planning Processes	• **Create a Project Management Plan (PMP)** and kept it up-to-date as the project progresses • PMP contains details to **monitor and exercise control over the project execution** (e.g. project schedule, milestones, project team members and all other project related information)
Executing Processes	• Once plan is in place, **execute the project** • Project Manager **evaluates adherence to project planning process, work products and services of the process to applicable requirements, objectives and standards**. Then, address noncompliance • Output: work results (e.g. proposal, working software, report)
Monitoring & Controlling Processes	• During project execution, Project Manager **monitor the progress against the plan** (e.g. check whether the project progress same as what has been planned, is tasks completed on time) • **Take corrective actions to solve identified issues**, may involve re-planning and establishing new agreements • **Output: overall health of project** (e.g. healthy if project is on time)
Closing Processes	• **When the project is completed**, Project Manager gathers project related data and documents • **Handover the project's product to customer** (e.g.

	completed software, user manual, operating manuals) • **Conduct post project review (self-reflection)** • **Redeploy project resources**

Cost & Benefit Analysis

Cost	Development Cost	Conduct feasibility study
		Consultation (e.g. consultation with security experts)
		Training
	Operating Cost	Power supply
		Stationery
		Network bill
Benefit	Tangible	Increase in savings
		Decrease in operating costs (e.g. Amount of money saved as result automation)
		Increase in revenue (more sales)
	Intangible	Employees' morale
		Better reputation
		Customers' loyalty

Project Appraisal and Evaluation

Net Profit

$$Net\ profit = total\ income - total\ costs$$

Year	*Project 1*	Project 2	*Project 3*	Project 4
0	*-100,000*	-1,000,000	*-100,000*	-120,000
1	*10,000*	200,000	*30,000*	30,000
2	*10,000*	200,000	*30,000*	30,000
3	*10,000*	200,000	*30,000*	30,000
4	*20,000*	200,000	*30,000*	30,000
5	***100,000***	**300,000**	*30,000*	**75,000**
Net Profit	***50,000***	**100,000**	***50,000***	**75,000**

Advantages

- **Easy to calculate** and intuitive

Disadvantages

- NP technique does **not consider the time value of money** (The value of dollar received in the future is not the same as a dollar received today)
- The situation of **long or late income return is not considered**. (Although Project 1 and 3 have the same net profit, Project 1 needs to wait longer for the return in Year 5 compared to Project 3 with consistent return)

Payback Period

- Calculate the **time needed to break even** or **pay back the initial investment**
- Normally project with the **shortest payback period will be chosen**
- Subtract each individual annual cash inflow from the initial cash outflow, until the payback period is achieved

Year	Project 1	Project 2	Project 3	Project 4
0	-100,000	-1,000,000	-100,000	-120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	**30,000**	**30,000**
5	**100,000**	**300,000**	30,000	75,000
Net Profit	**50,000**	***100,000***	**50,000**	***75,000***

Payback period for Project 1: Year 5

$-100000 + 10000 + 10000 + 10000 + 20000 + 100000 = 50000 \geq 0$

Payback period for Project 2: Year 5

$-1000000 + 200000 + 200000 + 200000 + 200000 + 300000 = 100000 \geq 0$

Payback period for Project 3: Year 4 (breakeven 收支平衡 in early Y4)

$-100000 + 30000 + 30000 + 30000 + 30000 = 20000 \geq 0$

Payback period for Project 4: Year 4 (breakeven at the end of Y4)

$-120000 + 30000 + 30000 + 30000 + 30000 = 0 \geq 0$

∴ Project 3 has the shortest payback period

Advantages

- **Easy to calculate** and understand
- Projects with **shorter payback periods are generally less risky** (as they recover their investment faster, reducing exposure to uncertainties)

Disadvantages

- It **ignores any profits generated after the payback period** (e.g. compare Project 3 and 4)
- PP technique does **not consider the time value of money** (it does not discount future cash flows)

Return on Investment (ROI)

- A.k.a. Accounting Rate of Return (ARR)
- **Compare the net profitability to the investment required**
- Formula:

$$ROI = \frac{Average\ Annual\ Profit}{Total\ Investment} \times 100$$

or

$$ROI = \frac{NPV}{Discounted\ Cost} \times 100\%$$

Year	Project 1	Project 2	Project 3	Project 4
0	-100,000	-1,000,000	-100,000	-120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000
Net Profit	**50,000**	**100,000**	50,000	75,000
ROI	?	?	?	?

Table 1.3 Four project cash flow projections – figures are end of year totals (RM)

Project 1:

$$ROI = \frac{\frac{50000}{5}}{100000} \times 100 = 10\%$$

Project 2:

$$ROI = \frac{\frac{100000}{5}}{1000000} \times 100 = 2\%$$

Project 3:

$$ROI = \frac{\frac{50000}{5}}{100000} \times 100 = 10\%$$

Project 4:

$$ROI = \frac{\frac{75000}{5}}{120000} \times 100 = 12.5\%$$

∴ Project 4 is the most worthwhile project based on ROI alone.

<u>Advantages</u>

- **Simple and easy to calculate**
- **Return rate provides a clear indicator of the potential returns**

Disadvantages

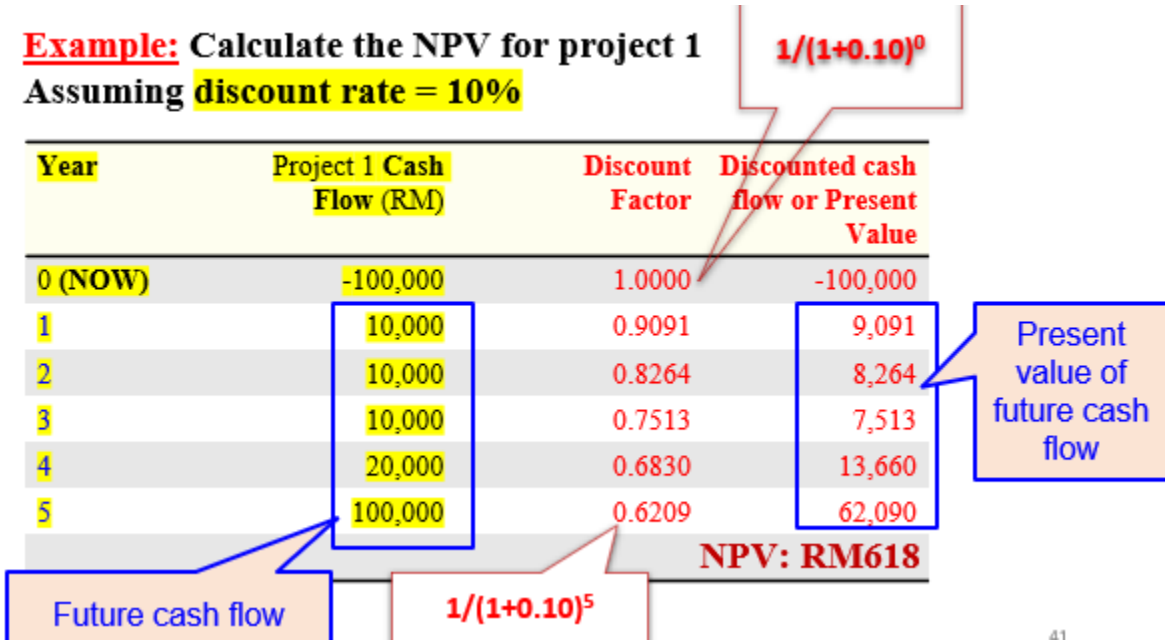
- Does **not consider the time value of money**
- **High ROI but a long payback period may not be good**

Net Present Value (NPV)

- NPV **considers the time value of money**
- It is a financial method used to **determine the current / present value of all future cash flows generated by a project** including the initial capital investment
- Formula:

  $$Present\ Value\ of\ a\ Future\ Cash\ Flow\ =\ Cash\ Flow\ in\ Year\ t\ \times \frac{1}{(1+r)^{t}}$$

  where
  - r = discount rate (expressed as a decimal value)
  - t = number of years into the future that the cash flow occurs



Year 0

$$Discount\ Factor\ =\ \frac{1}{(1+0.10)^{0}}\ =\ 1.0000$$

$$Discounted\ Cash\ Flow\ =-\ 100000\ \times\ 1.0000\ =-\ 100000$$

Year 1

$$Discount\ Factor = \frac{1}{(1+0.10)^1} = 0.9091$$

$$Discounted\ Cash\ Flow = 10000 \times 0.9091 = 9091$$

Year 2

$$Discount\ Factor = \frac{1}{(1+0.10)^2} = 0.8264$$

$$Discounted\ Cash\ Flow = 10000 \times 0.8264 = 8264$$

Year 3

$$Discount\ Factor = \frac{1}{(1+0.10)^3} = 0.7513$$

$$Discounted\ Cash\ Flow = 10000 \times 0.7513 = 7513$$

Year 4

$$Discount\ Factor = \frac{1}{(1+0.10)^4} = 0.6830$$

$$Discounted\ Cash\ Flow = 20000 \times 0.6830 = 13660$$

Year 5

$$Discount\ Factor = \frac{1}{(1+0.10)^5} = 0.6209$$

$$Discounted\ Cash\ Flow = 100000 \times 0.6209 = 62090$$

$$NPV = -100000 + 9091 + 8264 + 7513 + 13660 + 62090 = 618$$

∴ Project 1 has a positive NPV of 618 which indicates that the project is profitable (PV of cash inflows [benefits] > PV of cash overflows [costs] over the 5-year period)

Another method of calculating ROI

Exercise: Using a discount rate of 9%, calculate the NPV and **ROI** for Project B.

ROI = NPV / Discounted Cost x 100%

Project B	Year 0	Year 1	Year 2	Year 3	Total
Costs	100,000	60,000	60,000	50,000	
Discount Factor	1.0000	0.9174	0.8417	0.7722	
Discounted Costs	100,000	55,044	50,502	38,610	244,156
Benefits	0	80,000	90,000	180,000	
Discount Factor	1.0000	0.9174	0.8417	0.7722	
Discounted Benefits	0	73,392	75,753	138,996	288,141
Discounted benefits – costs	(100,000)	18,348	25,251	100,386	NPV = 43,985
~~Cumulative Discounted benefits – costs~~	~~(100,000)~~	~~(81,652)~~	~~(56,401)~~	~~43,985~~	-

ROI = Total Discounted Benefits – Total Discounted Cost / Total Discounted Cost x 100%
= (43,985/244,156) X 100% = 18.02%

46

Weighted Scoring Model

- **Specify the relevant criterion for projects selection**
- **Weight** the importance of each criterion.
- NPV, ROI can be the criterion.

Assign a score (0 to 100) to each criterion

- **High score** means the project strongly meet the specified criterion
- **Low score** means it is less likely to meet the specified criterion

Criteria (for project selection)	Weight (%)	Project 1	Project 2	Project 3
Support business objectives	10%	80	90	50
Availability of resources	10%	70	70	50
Duration < 12 months	10%	70	90	50
Low risk	5%	50	50	50
High probability of project success	15%	90	90	50
Good ROI (above 40%)	50%	80	90	50
Weighted Project Scores	100%	78	86	50

Weighted Project Scores for Project 1:

$(10\% \times 80) + (10\% \times 70) + (10\% \times 70) + (5\% \times 50) + (15\% \times 90) + (50\% \times 80) = 78$

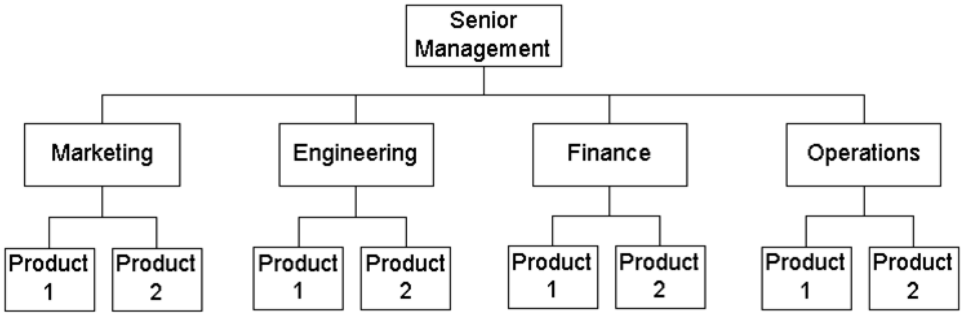
Weighted Project Scores for Project 2:

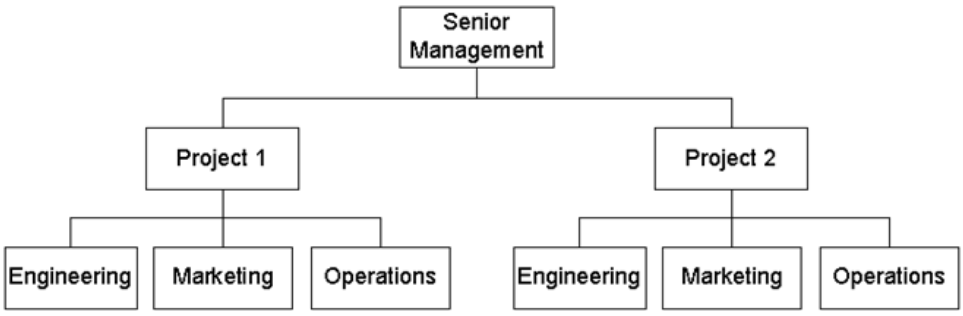
$(10\% \times 90) + (10\% \times 70) + (10\% \times 90) + (5\% \times 50) + (15\% \times 90) + (50\% \times 90) = 86$

Weighted Project Scores for Project 3:

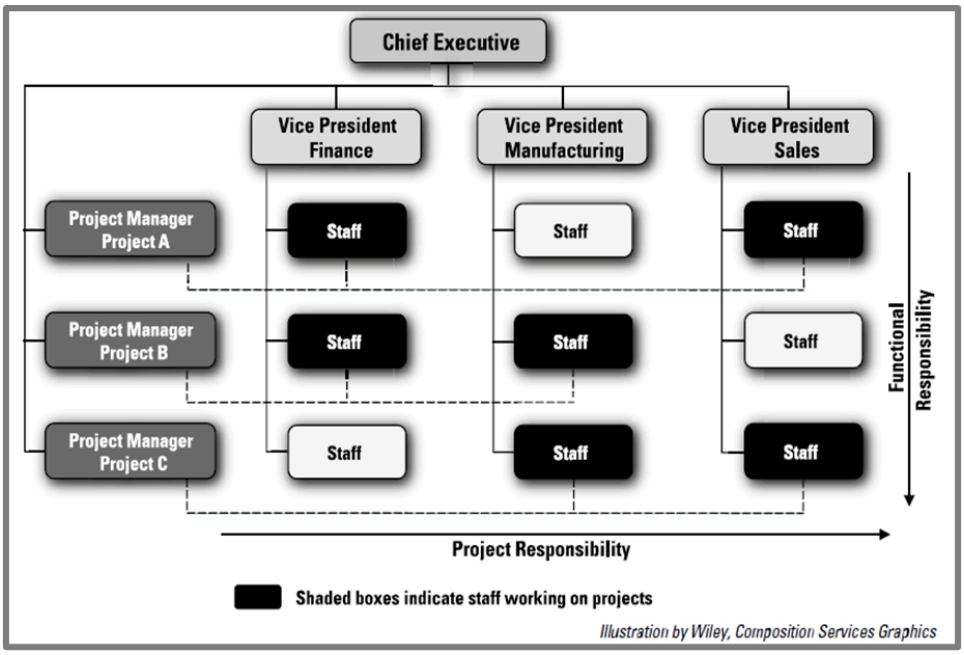
$(10\% \times 50) + (10\% \times 50) + (10\% \times 50) + (5\% \times 50) + (15\% \times 50) + (50\% \times 50) = 50$

Organization structures

Functional Organizational Structure	 ● Using an existing organization structure to manage project (groups employees based on specialized roles or functions such as marketing, finance, engineering) ● Project will be handled within the existing departments without creating a separate project team ● Pros ○ **Cost savings** ■ **No need for new teams or infrastructure**: Projects are executed using existing staff and resources, avoiding the costs of hiring or setting up dedicated project teams ■ **Efficient use of expertise**: Functional departments already have specialized knowledge, reducing the need for training or onboarding ■ **Lower overhead**: Administrative and managerial overhead is minimized since the project is embedded within the existing hierarchy ○ **Clear hierarchy and roles**: employees know exactly who they report to and what their responsibilities are ○ **Deep expertise**: Functional teams develop strong expertise in their areas, which can lead to

	high-quality work when their skills are applied to projects ○ **Easier resource allocation within departments**: Managers can assign tasks based on known capabilities and availability within their teams ● Cons ○ **Project leaders have little power**, project managers lack formal authority over team members, who report to their functional managers ○ **Conflicts of priorities**, staff are primarily focused on their departmental duties, so project tasks may be seen as secondary (e.g. transfer equipment to new office is not the staffs regular jobs) ○ **Slow decision-making**: Decisions may require approval from multiple functional managers, leading to delays ○ **Lack of project focus**: Projects may not receive the attention they need because they are not the primary focus of any one department, the employees may prioritize departmental goals over project goals.
Project Organizational Structure	 ● Project managers report to the CEO / Sr Mgmt ● Organization is arranged around projects rather than functions ● Project manager has full authority over the project and its team members ● Pros ○ **Unity of command** ■ Each team member reports to only one manager at a time, reducing confusion and conflict

	■ Project manager has full control over planning, execution and decision-making, which streamlines communication and accountability ○ **Highly responsive to project's objectives / customer needs** ■ Teams are dedicated to the project, allowing them to quickly adapt to changes in customer requirements or market conditions ■ Decisions can be made rapidly without needing approval from multiple departments ● Cons ○ **Duplication of facilities** ■ Each project may require its own equipment, tools and support systems, leading to duplication across projects (e.g. multiple projects might each purchase similar software licenses or set up separate testing labs, increasing overall costs) ○ **Cost inefficiency of resources** ■ Specialized staff or equipment may be idle when not needed by the project, leading to inefficiencies ■ Maintaining separate teams and infrastructure for each project can be expensive, especially for smaller organizations ■ Resources are not shared across departments, which can drive up costs

Matrix Organizational Structure	 ● Combination of functional and project structures ● Project managers often have staffs from various functional areas working on their projects ● Employees work across teams and report to two managers: functional manager and project manager ● Run multiple projects simultaneously and need to share resources efficiently ● Pros ○ **Efficient use of resources** ■ Staff from different departments can be assigned to multiple projects, maximizing use of specialized skills ■ Resources can be shifted between projects based on priority and workload ■ Avoids duplication of roles and facilities, unlike projectized structures ○ **Enhanced collaboration** ■ Encourages cross-functional teamwork and knowledge sharing ■ Promotes innovation by bringing diverse perspectives together ○ **Balanced focus** ■ Employees maintain their functional

	responsibilities while contributing to project goals ■ Helps align project outcomes with organizational strategy ● Cons ○ **Dual reporting (One staff reports to 2 managers / bosses)** ■ Employees may receive conflicting instructions from their functional and project managers ○ **Complex management** ■ Requires strong coordination and communication between managers ■ Decision-making can be slower due to the need for consensus 共识 between multiple leaders ○ **Potential for power struggles** ■ Managers may compete for control over resources or staff time ■ Lead to tension 关系紧张 and inefficiencies if roles and responsibilities are not clearly defined

Tools & Techniques for Project Team Acquisition

Resource assignment	● To **assign particular personnel to their projects** or to **acquire additional human resources needed to staff the project** ● PM with strong influencing & negotiating skills are often good at getting internal people to work on their projects ● Main outputs: project staff assignments, resource availability information, updates to the staffing management plan, project team directory
Resource loading	● **Amount of individual resources an existing schedule requires during specific time periods** ● Help PM and individuals to **develop schedules** ● Overallocation ○ more resources than are available are assigned

	to perform work at a given time. ○ e.g. Individual have to work 24 hours a day to meet the allocated schedule
Resource leveling	● A **technique for resolving resource conflicts (overallocation & underallocation)** ● To **create a smoother distribution of resource usage** ● PM examine the PERT chart for areas of slack or float, and to identify resource conflicts ● e.g. Remove overallocations by delaying non-critical tasks, which does not result in an overall schedule delay ● Resource leveling aims to **minimize period-by-period variations in resource loading by shifting tasks within their slack allowances**

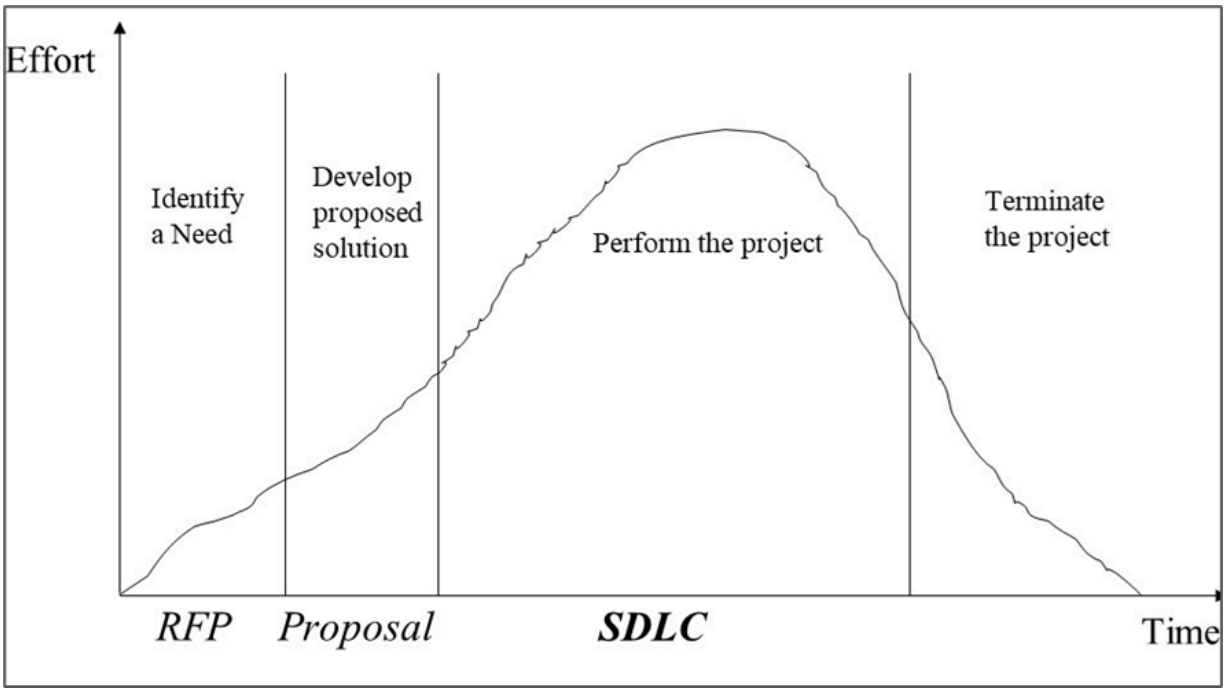
Project Management Failure

Project Objectives ● Is the **targets that the project team is expected to achieve** such as delivering: ○ Agreed functionality ○ Required level of quality ○ On time and within budget ● **Tactical and short-term** ● Focused on **deliverables, milestones and outcomes** of a particular project ● Usually defined by **project team or project manager**
Business Objectives ● For all the resources and efforts put in, **Return generated > costs** ● **Broad and long-term** ● Focused on **growth, profitability, market position or customer satisfaction** ● Set by **executives or leadership teams**

3 Scenarios

1. A project **meet the project objectives** and **meet the business objectives**
2. A project **meet the project objectives** but **fail to meet the business objectives**
3. A project **fail to meet the project objectives** but successfully **meet the business objectives**

Causes of Project Failure



- Does **not satisfy the stakeholders requirements**
- Does **not meet the project objectives (time, budget, quality)**
- Level 1: **Failures in Project Management Context**
    - Inappropriate "fit" of project organization to project objectives, project tasks, top management and environment at large
    - **Not using appropriate project process model**
        - e.g. Prototyping used in a large and complex system development which cannot capture the user requirements effectively
    - **Lack of top management support**
        - e.g. Lack of support from the financial department - hard to get funding from the organization
        - e.g. Lack of support from sales and marketing departments -

no personnels to promote the new games software

- Level 2: **Failures in Project Management System**
    - Include **wrong project manager**, neglect of systems approach in the project life cycle, and misuse of project management techniques
    - e.g. Project manager is unable to control conflicts
    - e.g. Hardware, software, resources and facilities are viewed independently without relation to their overall project objectives - poor resource planning during the 3rd stage (Perform the project) where involves the most people to do the work
- Level 3: **Failures in Planning and Control Processes**
    - **Inadequate communication, planning, poor estimation, poor scheduling or lack of user involvement**
    - e.g. Information about scope, objectives, responsibility and acceptance criteria not documented early in the project
    - The behavior of the project team discourages user involvement
- **Project Manager too optimistic**
- **Poor team development**
- **Under-allocation of resources** (insufficient resources such as people, equipment or budget) allocated to a project activity
- To achieve its objectives within the specified timeframe and quality
- Organization culture (e.g. no teamwork or no sharing of information between employees)
- External factors (e.g. government enforce new laws)

How to reduce business risks?

- Carry out **market surveys** before development
- **Competitor analysis**
- **Prototyping and evaluation by potential users**

Critical Success Factors (CSFs)

- Make sure you have the support of top management
- Manage the project scope effectively
- Competent and experienced project teams

Business Goals & Project Goal

	Project Goal	**Business Goal**
Scope	Narrow, specific to a project	Broad, organization-wide
Timeframe	Short-term (weeks to months)	Long-term (months to years)
Purpose	Deliver a specific output or result	Achieve strategic business objectives
Measurement	Based on project deliverables	Based on business performance metrics
Examples	• Develop a new product (Launch a new mobile banking app within 6 months) • Implement a system upgrade (Migrate all customer data to new CRM system by Q4) • Run a marketing campaign (Execute a social media ad campaign to generate 5,000 leads in 3 months) • Improve internal processes (Reduce invoice processing time by 40% by automating workflows within 6 months)	• Increase revenue (Grow annual revenue by 20% within the next 3 years) • Expand market share (Capture 10% of the Southeast Asian e-commerce market by 2026 • Improve customer satisfaction (Achieve a Net Promoter Score (NPS) of 70+ by the end of next year) • Enhance brand awareness (Increase brand recognition by 30% through digital campaigns in 12 months)

C2: Project Planning, Control, Process Models

Stakeholders

Category	Description
Project team	They are directly involved in doing the project work
External to project team but within the same organization	e.g. Project leader might need the assistance of the users to carry out system testing
External to both project team and organization	e.g. • **Contractors** who will carry out specific works for the project • **Suppliers** who supplies hardware • **Customers** who will benefit from the system that the project implements

SMART

Criteria	Description
Specific	The project objective should be clear, well-defined and unambiguous *e.g. To achieve minimum customer satisfaction rating of 8/10 by 30-Jun*
Measurable	There should be measures of effectiveness which tell us how successful is the project *e.g. Measure customer satisfaction level*
Achievable	It must be within the power of the individual or group to achieve the objective

	e.g. Customer satisfaction target = 8/10 is achievable?
Relevant	The objective must be relevant to the true purpose of the project *Project is something that you will carry out* *Objective is the desired outcome that you want to achieve*
Time constrained	There should be a defined point in time by which the objective should have been achieved *e.g. 30-Jun is the deadline to achieve the desired outcome*

Sample

To implement a Hotel Room Reservation System by 01-01-2026 that will reduce booking time to 10 minutes at a cost not more than $100,000.

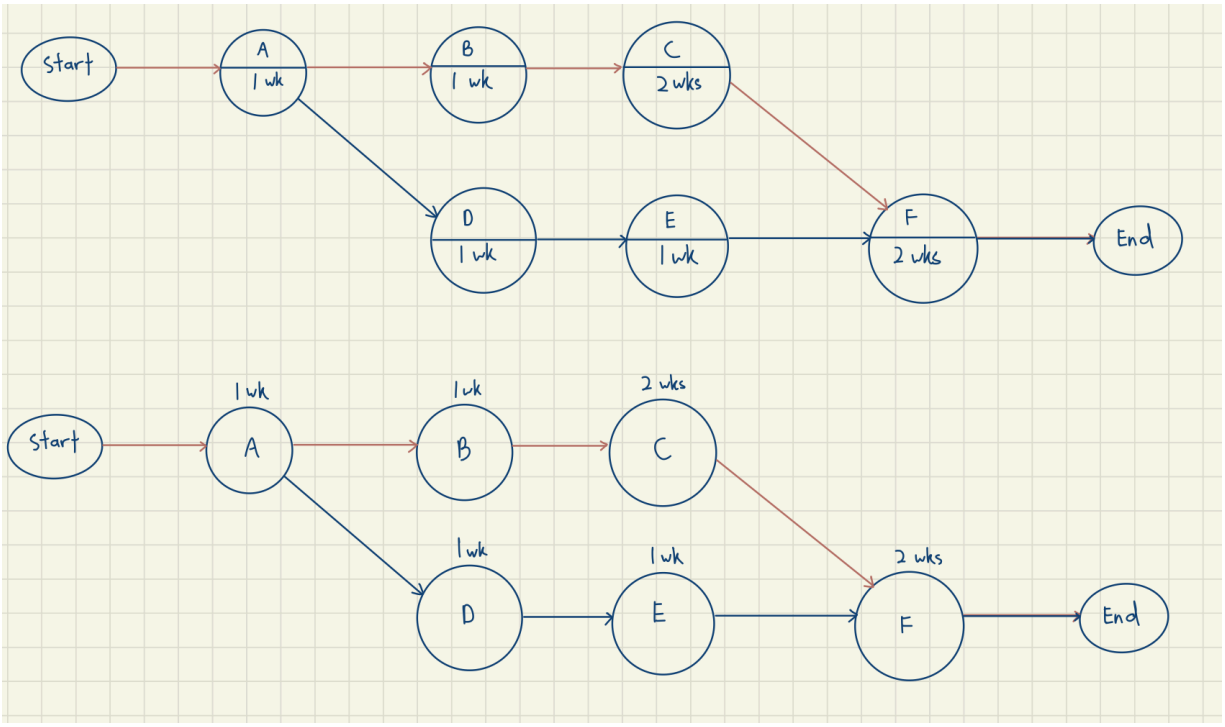
Specific	"To reduce booking time to 10 minutes at a cost not more than $100,000 by 01-01-2026" is clear, well-defined and unambiguous
Measurable	The objective has stated the amount of time taken to complete the project (by 01-01-2026), amount money spent to complete the project (not more than $100,000) and time user spent to complete room booking (10 minutes)
Achievable	The objective has stated that the system will be able to reduce the booking time to 10 minutes within the $100,000 budget once it has been completed.
Relevant	The objective of developing the Hotel Room Reservation System is relevant to the purpose of reduce booking time to 10 minutes.
Time constrained	The objective has stated that the project should be done by 01-01-2026.

✅Program Evaluation and Review Technique (PERT)

1. Identify specific activities and milestones
2. Determine the sequence of activities

3. Estimate the time for each activity
4. Construct a network diagram
5. Identify the critical path

Task ID	Task Name	Duration (in week)	Predecessor
A	Collect requirements	1	None
B	Design online module	1	A
C	Code online module	2	B
D	Design offline module	1	A
E	Code offline module	1	D
F	Testing online and offline module	2	C, E



- The critical path is A - B - C - F.
- The total duration needed to complete the project is 6 weeks = 1 week + 1 week + 2 weeks + 2 weeks.

Project Monitoring and Control

Tracking the Schedule

- Conduct **periodic project status meetings** (each team member reports progress and problems)
- **Compare the actual start date against the planned start date** for each project task listed in the project table
- **Check whether formal project milestones have been accomplished by the scheduled date**
- **Evaluate the results of all reviews conducted** throughout the software engineering process
- Use **Earned Value Analysis to assess progress quantitatively**

✅Corrective Actions

- Project manager monitors project progress in terms of time, cost and quality (triple constraints)

Corrective Action	Description

Adding more staff	• Applicable when **behind project schedule**
	• Will work if a **task can be partitioned** (interviewing users to gather requirements)
	• Cons:
	  ○ Will **incur additional cost**
	  ○ May further delay the task because need to **consider communication, learning curve, time to get-up to speed**, etc

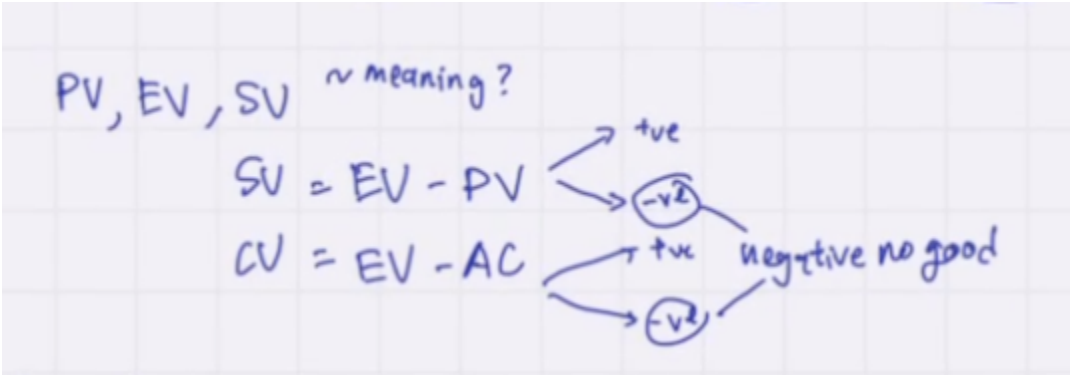
Adding different skills	• Adding people with different or greater skills • When **problems need skill / knowledge** or **staff lack experience**, adding experienced or skilled staff to a team may yield results (e.g. adding a person who skillful in UI design)
Reassigning Task	• No need to add staffs or skills • Simply **switch tasks around as each member has different strength** • may **result in higher productivity or better quality** of work as different staffs may have different potential and knowledge at their respective areas
Increasing individual supervision	• **Partitioning tasks** and **creating smaller deliverables** to exercise **quality control more frequently** • Allow inexperienced / less confident staff to **obtain guidance**
Decreasing individual supervision	• **Experienced staff may resent 不满 when dealing with too frequent check up** • May **affect personal interest and work quality** • Reduces work of supervision and re-channel 重新引导 resourceful focus
Improving methods	• Example: Using Joint Application Development (JAD)

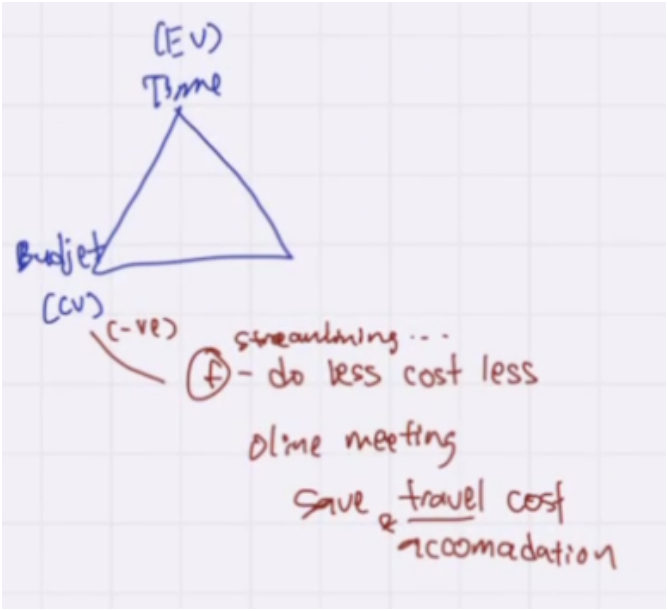
of working	approach to reconcile what seemed to be mutually exclusive requirements between different users, staging workshop where all users can come together to thrash out the differences during analysis stage 采用联合应用程序开发(JAD)方法，协调不同用户之间似乎相互排斥的需求，举办研讨会，让所有用户在分析阶段共同消除分歧
Streamlining the process	• **Make the process more efficient or simplified the process** • Some tasks like quality control may be bureaucratic 繁琐 and time-consuming • Will be aggravated 更糟糕 if the procedures or processes are vague 模糊 • Streamline processes by **removing unnecessary activities** and **using standardized forms and process** (e.g. using standardised charge request form, down C,R, form from the intranet)
Changing resource priorities	• **Examine project critical path** to **relocate the priorities of access**
Re-planning the project	• **Rework the plan** • Make the **task dependencies clearer** after the work has started • Need to **note while new plans remove problems of old plan**
Changing the phasing of deliverables	• Changing the phasing of the deliverables can **improve the effectiveness** • May consider **phased delivery to focus on urgent requirements** when dealing with discrete elements that have higher priorities • May also consider **parallel working** (e.g. design overlap with analysis)
Decreasing the number of inspections	• Applicable only if the **inspection (assessment) are uncovering an acceptably low number of defects**

	● Else **problems will arise later** and does not result in any positive effects
Increasing the number of inspections	● Applicable when they are **critical systems** or **intermediate deliverables are less satisfactory** or there is a **high level of defects on completed deliverables** ● Ensure **errors are discovered earlier and rectified** more quickly ● Cons: ○ May **delay the project** when number of inspections increases
Encouraging the team	● **Project fatigue** may cause **lower productivity**, **increased absenteeism, resignations and complaints** ● Action can be taken: ○ **Refocusing on team's achievements** to rekindle enthusiasm 重新关注团队成就，重燃热情 ○ **Organize social events in company** time to engender a sense of team spirit 利用公司时间组织社交活动，培养团队精神 ○ **Redistribution of work** to provide development opportunities ○ A **team building exercise** of some sort ○ **Reducing the size of deliverables** (at the end may be larger)
Introducing incentives	● Incentives may be in forms: ○ **Time off** 休息时间 ○ **Recognition** 表彰 ● For multiple team, **inter-team competition** may be considered but practiced with care
Subcontracting part of the work	● Project manager may consider **sub-contracting out the tasks to those with special skills / facilities** 项目经理可考虑将任务分包给具有特殊技能/设施的人员

	• Make sure the **contractors work to the required standards**
Negotiating changes in the specification	• When all else fail, may **negotiate to change the specification** (may be the original objective is too ambitious for the time or money) • Alternatively, may resort to **phased delivery with major functions being delivered first**

✅Earned Value (EV) Analysis





- EV analysis is a refinement of cost monitoring (consider the project progress and schedule)
- Helps to **check whether the project cost has exceeded its budget, or behind schedule or ahead of schedule**.
- EV Analysis is based on a "value" to each task based on original

expenditure forecasts. This assigned value is called planned value (PV).

- PV is equivalent to the price agreed upon by a contractor to do the unit of work
- EV = actual value of work completed (EV = PV when a task has been completed)

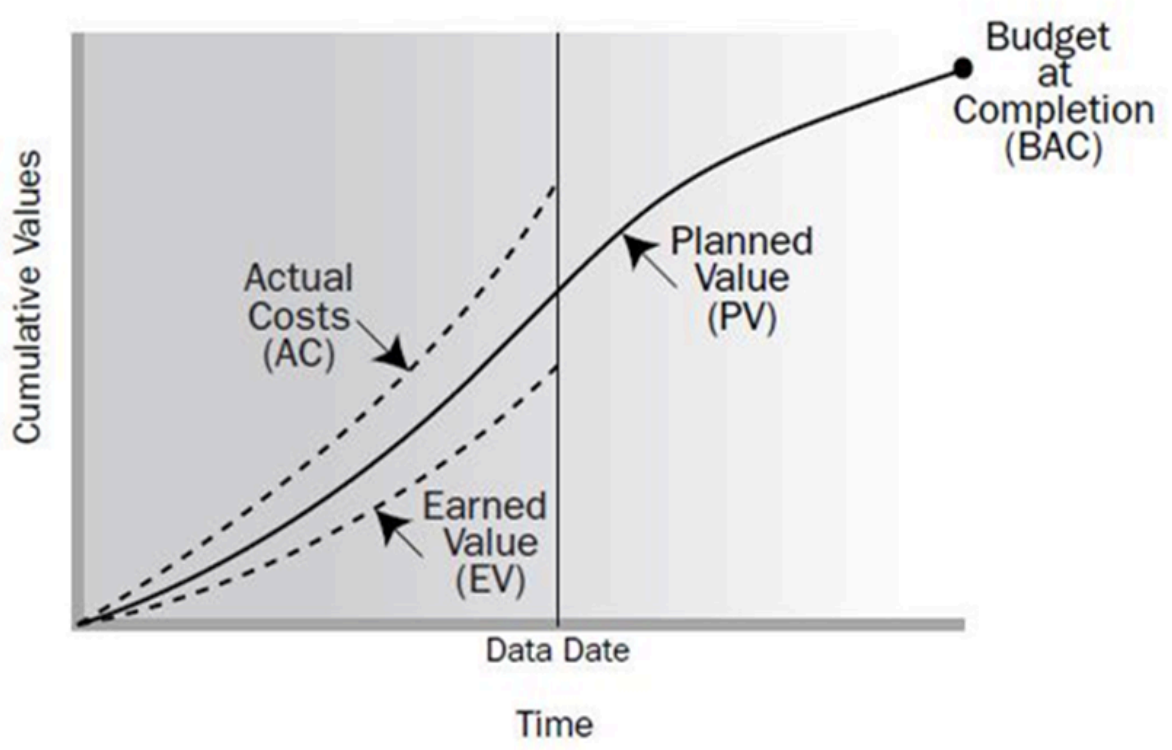
Techniques for Crediting or assigning EV to a project:

- 0/100 technique: EV = 0 when job started, EV = PV when job is finished (assign 100% of PV to EV when the work is finished)
- 50/50 technique: EV = 50% when job started, EV = PV when job is finished (add remaining 50% of PV to EV when the work is finished)
- 75/25 technique: EV = 75% when job started, EV = PV when job is finished (add remaining 25% of PV to EV when the work is finished)
- milestone technique: a task is given a value based on the achievement of milestones that have been assigned values as part of the original budget plan.
- % complete: the EV is based on the % of the project that has been completed (assign the value to EV based on percentage of work completed)

Steps

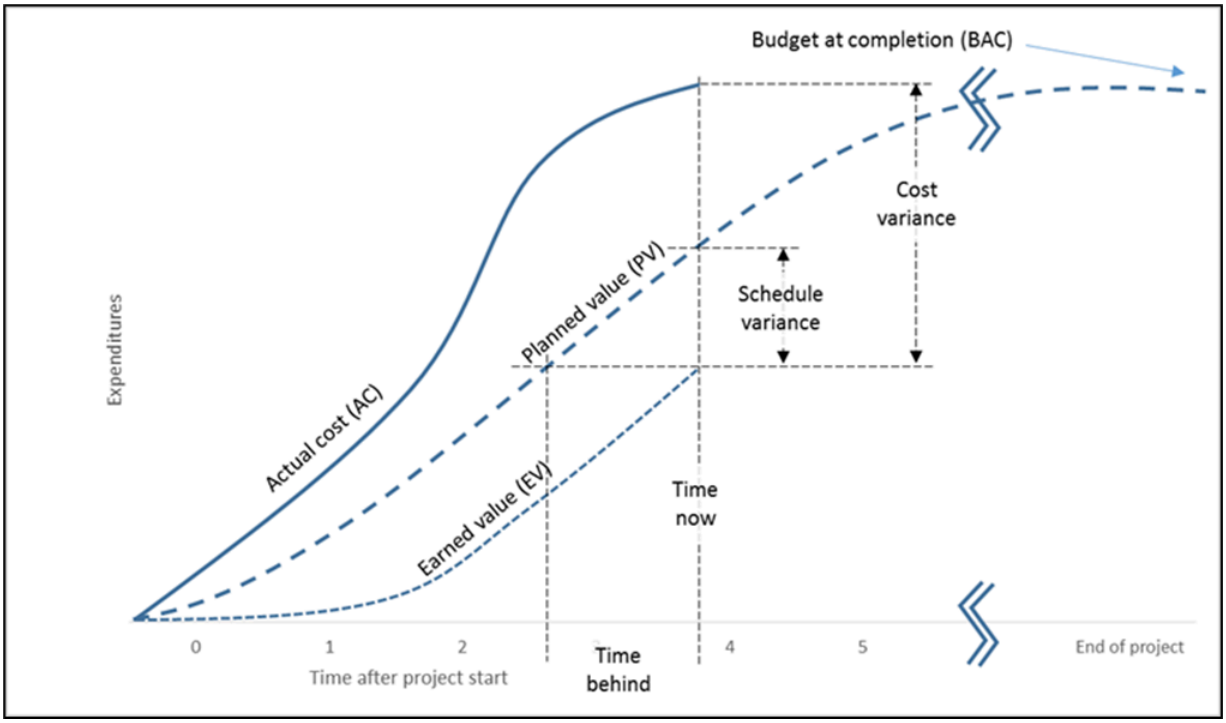
1. Create the baseline budget
2. Monitor the earned value
3. Calculate Performance Statistics (after knowing PV and EV)

Monitor the earned value



Metric	Description
Planned Value (PV)	• **Value of the work planned to be completed** • PV = Sum of estimated cost (until the specified time)
Budget at Completion (BAC)	**Budget for the whole project**
Earned Value (EV)	• **Actual value of the work completed** • EV = Sum of [Estimated Cost * Status] (not consider specified time, include all task as long as the task status is more than 0%) • EV = BAC × % of works completed
Actual Cost (AC)	**Actual cost incurred for the work completed**

Calculate Performance Statistics



Metric	Description
Cost Variance (CV)	• CV = EV - AC • **Negative** indicates **cost has exceeded its budget** (at time t, amount paid is > actual value of work completed) • **Positive** means **cost is within the budget** (at time t, amount paid is < actual value of work completed)
Schedule Variance (SV)	• SV = EV - PV • **Negative** indicates **project is behind schedule** (at time t, actual value of work completed is < value of work planned to be completed) • **Positive** means **project is ahead of schedule** (at time t, actual value of work completed is > value of work planned to be completed)

Sample

Exercise #1

You are now in the 9th month of 29 months project. As of today, you have spent RM198000 based on the invoices reconciliation. You need to provide data to Mr. Jordon (Project Sponsor), specifying if you are making a profit or loss in the Project. You are using EVM technique to provide the sponsor with this information.

Answer

Task	Estimated Duration (month)	Estimated Cost ($)	Status	EV
A	2	70,000	100%	70,000
B	3	27,000	85%	22,950
C	1.5	82,000	62%	50,840
D	2.5	25,000	48%	12,000
E	2	30,000	23%	6900
F	0.5	48,000	16%	7680
G	1.5	100,000	0	
H	2	10,000	0	
I	1	45,000	0	
J	2.5	17,000	0	
K	3	32,000	0	
L	1.5	12,000	0	
M	2	10,000	0	
N	3	24,000	0	
O	1	12,000	0	
Total	29	544,000	0	

PV

Actual value of work completed in the 9th month

You are now in the **9th month,**
AC (actual cost) = **198,000**
Budget At Completion (BAC) = **544,000**

PV = 70k + 27k + 82k + 25k = **204,000** (9 months)

EV = (70k*100%) + (27k*85%) + (82k*62%) + (25k*48%) + **(30k*23%)** + **(48k*16%)**
= 170, 370

SV (schedule variance)= EV - PV
=170,370 – 204,000
= -33,630

value of work **planned** to be completed in the 9th month

SV is negative means that the project is behind schedule *(In the 9th month, **actual** value of work completed is < value of work **planned to be completed**)*

Exercise #2 (using different set of data)

You are now in the 9th month of 29 months project. As of today, you have spent RM198,000 based on the invoices reconciliation. You need to provide data to Mr. Jordon (Project Sponsor), specifying if you are making a profit or loss in the Project. You are using EVM technique to provide the sponsor with this information.

Task	Estimated Duration (month)	Estimated Cost ($)	Status	EV
A	2	70,000	100%	
B	3	27,000	100%	
C	1.5	82,000	100%	
D	2.5	25,000	100%	
E	2	30,000	50%	
F	0.5	48,000	50%	
G	1.5	100,000	0	
H	2	10,000	0	
I	1	45,000	0	
J	2.5	17,000	0	
K	3	32,000	0	
L	1.5	12,000	0	
M	2	10,000	0	
N	3	24,000	0	
O	1	12,000	0	
Total	29	544,000	0	

PV

Actual value of work completed in the 9th month

You are now in the 9th month,
AC (actual cost) = 198,000
Budget At Completion (BAC) = 544,000

PV = 70k + 27k + 82k + 25k = **204,000** (9 months)

EV = **243,000**

SV (schedule variance)= EV - PV
= 243,000 – 204,000
= 39,000

value of work **Planned** to be completed in the 9th month

- SV is positive means tha the project is ahead of schedule *(In the 9th month, actual value of works completed is > value of works planned to be completed)*
- MORE works have been completed than planned!

65

Exercise #2 (using different set of data) (con't)

You are now in the 9th month of 29 months project. As of today, you have spent RM198.000 based on the invoices reconciliation. You need to provide data to Mr. Jordon (Project Sponsor), specifying if you are making a profit or loss in the Project. You are using EVM technique to provide the sponsor with this information.

Task	Estimated Duration (month)	Estimated Cost ($) (PV)	Status	EV
A	2	70,000	100%	
B	3	27,000	100%	
C	1.5	82,000	100%	
D	2.5	25,000	100%	
E	2	30,000	50%	
F	0.5	48,000	50%	
G	1.5	100,000	0	
H	2	10,000	0	
I	1	45,000	0	
J	2.5	17,000	0	
K	3	32,000	0	
L	1.5	12,000	0	
M	2	10,000	0	
N	3	24,000	0	
O	1	12,000	0	
Total	29	544,000	0	

You are now in the 9th month,
AC (actual cost) = 198,000

PV = 70k + 27k + 82k + 25k = 204,000 (9 months)

EV = 243,000

SV (schedule variance)= EV – PV = 39,000

CV (cost variance) = EV- AC
= 243,000 – 198,000 = **45,000**

Actual value of work completed

Actual amount paid

- CV is positive, it means that the project is under budget.

- In the 9th month, ***actual value of work completed is > amount paid***

- However, **if CV is negative,** it means that *amount paid is more than actual value of work completed!*

66

Exercise #3 (again using different set of data)

~~You are now in the 9th month of 29 months project. As of today, you have spent RM198.000 based on the invoices reconciliation.~~ You need to provide data to Mr. Jordon (Project Sponsor), specifying if you are making a profit or loss in the Project. You are using EVM technique to provide the sponsor with this information.

Task	Estimated Duration (month)	Estimated Cost ($) (PV)	Status	EV
A	2	70,000	100%	70,000
B	3	27,000	100%	27,000
C	1.5	82,000	100%	82,000
D	2.5	25,000	100%	25,000
E	2	30,000	100%	30,000
F	0.5	48,000	100%	48,000
G	1.5	100,000	100%	100,000
H	2	10,000	100%	10,000
I	1	45,000	100%	45,000
J	2.5	17,000	100%	17,000
K	3	32,000	100%	32,000
L	1.5	12,000	100%	12,000
M	2	10,000	100%	10,000
N	3	24,000	100%	24,000
O	1	12,000	0%	
Total	29	544,000	0	

* You are now in the 28th month
* AC (actual amt paid) = 544,000

PV = 70k + 27k + … + 24k = **532,000**

EV = 70k + 27k + … + 24k = **532,000**

CV (cost variance) = EV- AC
= 532,000 - 544,000
= -12,000

Actual value of work completed

Actual amount paid

- CV is negative, means that the project cost has exceeded its budget.

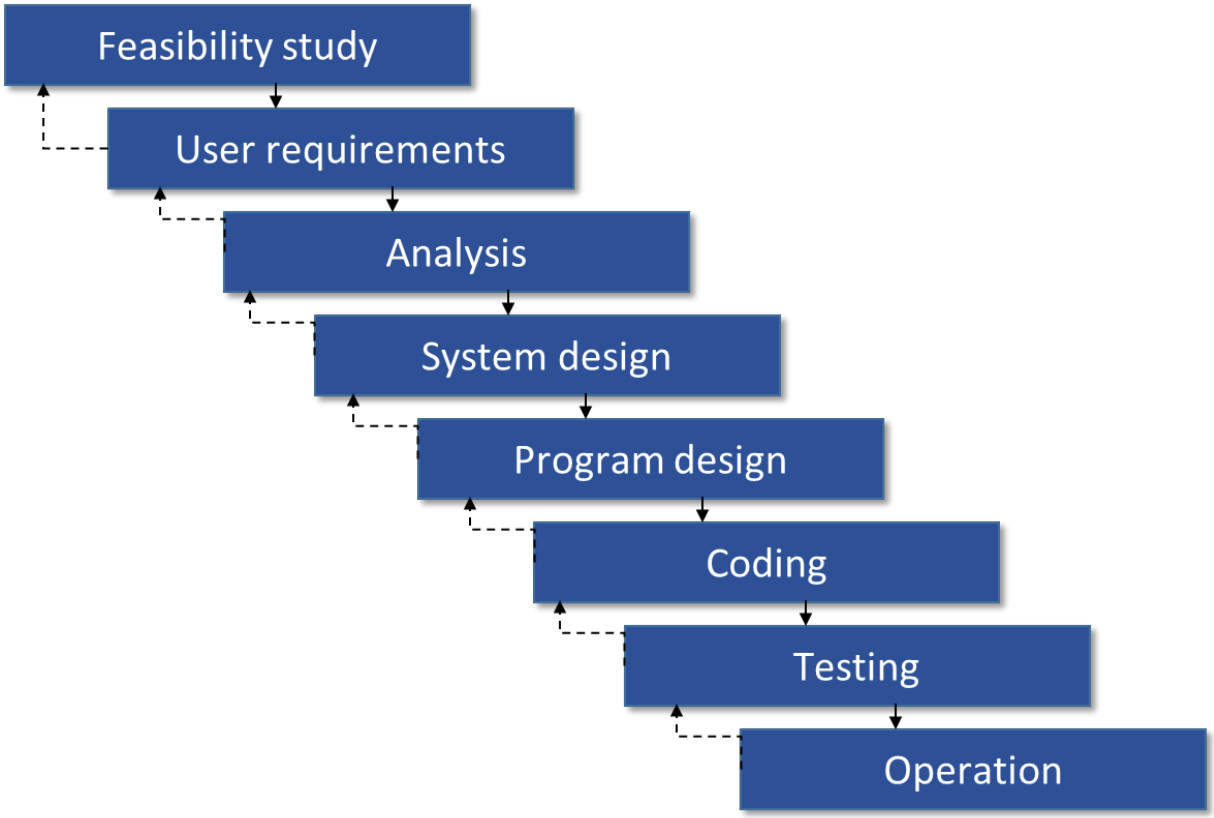
- On the 28th month, ***amount paid is more than actual value of work completed*** *(544K vs 532K)*

Q) How much more are needed to pay complete the project? 12K

Total amount to complete the project will be **556K**

Selection of Process Models

Waterfall Model



- 'Classical' model of system development
- There is a **sequence of activities working from top to bottom**
- The later stage may reveal the need for some extra work at an earlier stage.
- For large projects, try to avoid reworking tasks previously thought to be completed. 避免对之前认为已经完成的任务进行返工

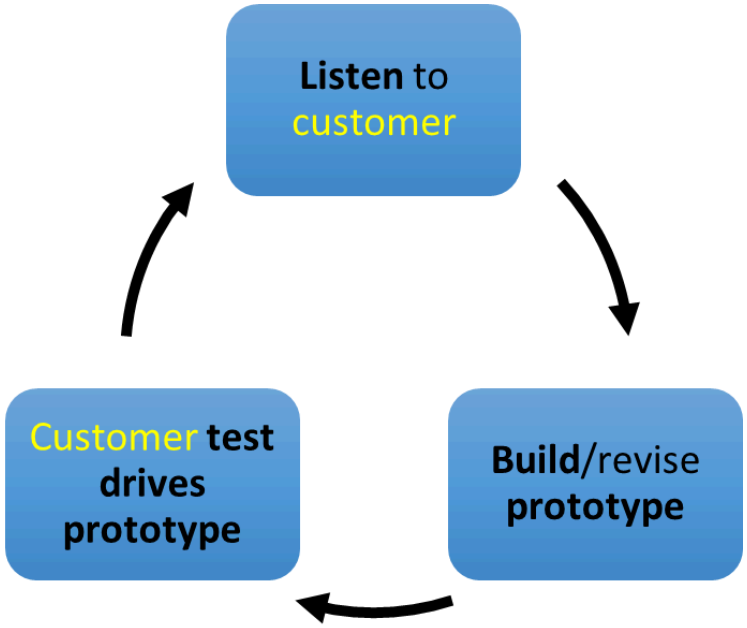
Pros:

- **Creates natural milestones at the end of each phase**
- Can **easily check on project progress**
- **Suitable** for project where **requirements are well defined** and **development methods are well understood**

Cons:

- **Not suitable for project with uncertainty** (not recommended when user **requirements are not well defined**)
- **Not flexible** (**reluctant to go back to previous stage**)

Prototyping Model



- **Evolutionary prototype**
  - The prototype is **built in stages**, with **each stage adding more features and refinements**
  - It will **evolve into the final system**
- **Throw-away prototype**
  - The prototype is **built rapidly to explore ideas and gather feedback**
  - It is **discarded after the necessary information is gathered**
  - **Not used in the final system**

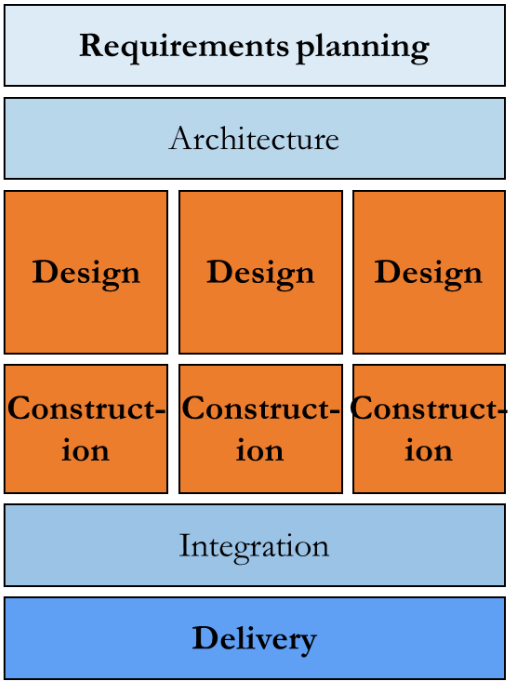
Pros:

- **Useful when user requirements are uncertain**
- **Functional / valuable in designing system's UI** (data-entry screen, reports or Web pages)
- **Encourages user's (customer's) involvement**

Cons:

- May **skip essential steps in system development** (e.g. testing phase)
- **Time-consuming** if several iterations are needed to **refine the design, lead to longer project timelines**

Rapid Application Development (RAD)



- Incremental software development process model that emphasizes **short development time**
- **System can be produced within 60 - 90 days**, if **requirements and project scope are well defined**
- **Each RAD team will take a major function/component/module** (component to be accomplished within 3 months) and then **integrated to form a whole**

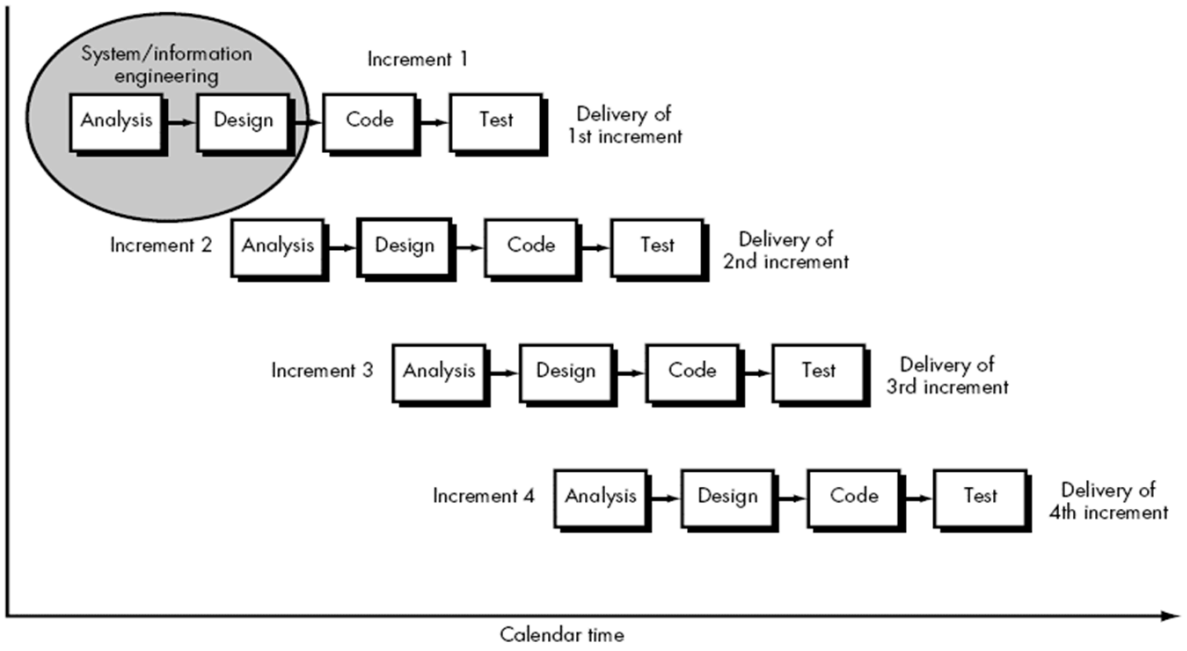
Pros:

- System can be **completed in a short period of time** (within 60 - 90 days) because **several RAD teams are developing the system simultaneously**
- **Easier to accommodate changing requirements** due to short iteration time spans

Cons:

- For large but scalable projects, RAD **requires large number of resources** to create the right number of RAD teams
- **Requires developers and customers committed to the rapid-fire activities** 要求开发人员和客户致力于快速活动
- **Not suitable for system that cannot be modularized**
- **Not suitable when technical risks are high** (e.g. New system use new technology, new software needs to integrate with existing system)

Incremental Model



- **Linear sequential model + prototyping**
- The **software is designed, implemented and tested in increments** (smaller portions or modules), not delivering the entire system at once
- **Each increment builds on those that have already been delivered**

Pros:

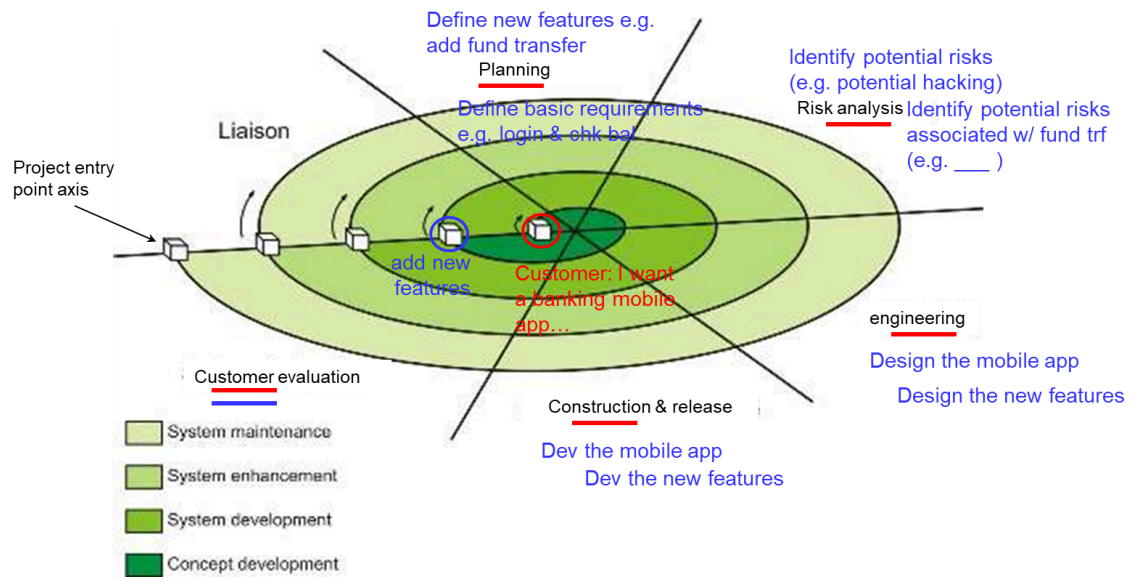
- **Easy for breakdown of tasks** because of divide and conquer approached used
- **Lowers initial delivery cost**

- **Can deduct errors easily** because core modules are used by the customer from the beginning of the phase and tested thoroughly
- **More flexible and less costly to change scope and requirements**
- **Easy to manage risk** because of iterations

Cons:

- **Requires good planning designing**
- **Definition of system should be complete and clear**
- **Increased complexity**: The project may become more complex as each increment is added. This makes it harder to manage and maintain, as well as increase the risk of errors and bugs.

Spiral Model



- System to be implemented is considered in more detail in each sweep in the loop
- **Each sweep terminates with an evaluation before the next iteration is embarked upon** 每次扫频结束时都要进行一次评估，然后再开始下一次迭代
- **Reduce risk** (lower risk means high chance for project success)

Pros:

- **Software is produced early in the software life cycle**
- Best development model to **handle risk since it has risk analysis and**

risk handling at every phase

- **Flexible and easy to change requirements at later phases** and can be incorporated accurately
- **Good for large and complex projects**
- **Suitable for high risk projects** where business needs may be unstable

Cons:

- **Not suitable for small projects as it is expensive**
- **More complex** than other SDLC models
- Too much **dependable on risk analysis and requires highly specific expertise**
- **Difficult to manage time** because the **number of phases is unknown at the start of the project**, time estimation is very difficult.
- **Not suitable for low risk projects**

✅Summary of Project Process Model Selection

Models	Project Scope	Development Time	Budget	Manpower	User Requirement	User
Waterfall	Large-scale system	Long	High	Many staffs	Well defined	Less involved
Prototyping	Small to medium	Flexible	Limited	Few staffs	Uncertain	Highly involved
Rapid Application Development (RAD)	System can be partitioned / modularized	3 months	High	Many staffs (e.g. each team to handle 1 module)	Well defined	Highly involved
Incremental	Flexible / complex	Short	Low	Lack of staff (few staffs)	Well defined	Highly involved
Spiral	Large-scale, risky system	Long	High	Need experts	Uncertain	Highly involved
Component-based	System that can be	Short	Low	Less staffs	Well defined	Highly

development	partitioned / componentized					involved

C3: Quality Management & Assurance

Software Quality

Importance	• Organizations are relying more on computer systems • Most safety-critical systems have built-in software, high quality software **ensure that system functions as expected in all situations** • Quality **retains customers and increases profits** • Quality is **essential for international marketing**
Definition	• A **product meet its specification** • The subjective quality of a software system is largely based on its non-functional characteristics. This reflects practical user experience

Software Quality Attributes (ISO/IEC 25010 Quality Model & Attributes)	• **Functional suitability**
	  ○ Functional completeness
	  ○ Functional correctness
	  ○ Functional appropriateness
	• **Performance efficiency**
	  ○ Time behavior
	  ○ Resource utilization
	  ○ Capacity
	• **Compatibility**
	  ○ Co-existence 共存
	  ○ Interoperability 互操作性
	• **Usability**
	  ○ Appropriateness recognizability
	  ○ Learnability
	  ○ Operability
	  ○ User error protection
	  ○ User interface aesthetics
	  ○ Accessibility
	• **Reliability**
	  ○ Maturity
	  ○ Availability
	  ○ Fault tolerance
	  ○ Recoverability

	• **Security**
	  ◦ Confidentiality
	  ◦ Integrity
	  ◦ Non-repudiation
	  ◦ Accountability
	  ◦ Authenticity
	• **Maintainability**
	  ◦ Modularity
	  ◦ Reusability
	  ◦ Analysability
	  ◦ Modifiability
	  ◦ Testability
	• **Portability**
	  ◦ Adaptability
	  ◦ Installability
	  ◦ Replaceability

✅Software Quality Management 3 Main Activities

Quality Assurance (QA)

Definition	• To **develop an organizational framework (procedures & standards) that will lead to high quality of software** • **Creating the framework, rules and environment** so that quality naturally happens throughout the project • QA establishes the infrastructure that supports solid software engineering methods, rational project management and quality control actions

Example	• Standard and procedures to be followed during design and development phases
	• Coding Guidelines
	  ◦ QA sets organization-wide coding standards (naming conventions, indentation, error handling)
	  ◦ Developers across all projects follow the same rules → consistent, maintainable code
	• Design Reviews
	  ◦ QA requires formal design review checklists before

	coding starts ○ Ensure architecture decisions are validated early ● Testing Frameworks ○ QA defines mandatory testing practices (unit tests, regression tests, automated CI/CD pipelines) ○ Every project uses the same baseline testing approach ● Documentation Standards ○ QA enforces IEEE or ISO documentation templates for requirements, design and test cases ○ Guarantee clarity and traceability across teams ● Process Models ○ QA decides whether the organization follows Agile, Waterfall or Hybrid models ○ Provides training and tools so teams can apply them consistently

Quality Planning (QP)

Definition	● To **select appropriate procedures and standards from the framework and adapt to a specific software project** ● **Deciding in advance how to ensure quality** in a software project ● **Pick the right standards, methods and procedures** (from frameworks like ISO, CMMI, Agile practices, etc) ● **Adapt them to your specific project** so the final product meets customer expectations
Example	● Based on the situation of current project, select the appropriate procedures and standards which has been defined in earlier phase to achieve the project goal ● Coding Standards Example ○ Use Java coding standards (naming conventions, error handling rules) and enforce peer code reviews ● Testing Standards Example ○ Apply ISTQB testing procedures and require unit + integration tests before deployment ● Documentation Example ○ Follow IEEE documentation standards for

	requirements and design ● Process Example ○ Use Scrum ceremonies (daily stand-ups, sprint reviews) and Definition of Done as quality gates ● Security Example ○ Apply HIPAA security standards and run penetration testing

Quality Control (QC)

Definition	● To **execute process to ensure that software development follows the quality procedures and standards** ● QC encompasses a set of software engineering actions that help to ensure that each work product meets its quality goals ● QC involves **monitoring the software development process** to **ensure that QA procedures and standards are being followed** ● The compliance of product to the defined project standards can be checked via: ○ **Quality reviews** ○ Automated software assessment
Quality Reviews	● Involve a group of people examining part or all of a software process, system or its documentation to discover potential problems ● Outcome of the review: ○ List of problems found ○ Author to do corrections ● Code Reviews (Quality Reviews) ○ A team examines a developer's code for bugs, style violations or missed requirements ○ Outcome: A list of issues → developer fixes them ● Automated Testing ○ Running unit tests, integration tests and regression tests automatically ○ Ensures new code does not break existing functionality ● Static Code Analysis ○ Tools like SonarQube and Lint scan code for

	vulnerabilities, complexity or style violations ○ Helps catch issues early ● System Testing ○ Running the software in a test environment to check performance, security and usability ○ Confirms the product works as intended ● Documentation Review ○ Reviewing requirement specs or design docs for completeness and consistency ○ Prevents misunderstandings before coding

QA vs QP vs QC (Quick Comparison)

Aspect	**Quality Assurance (QA)**	**Quality Planning (QP)**	**Quality Control (QC)**
Focus	Organization-wide framework	Project-specific plan	Execution & monitoring
Goal	Build infrastructure & rules	Choose / adapt standards for one project	Verify compliance & detect defects
When	Ongoing, across all projects	At project start	During development & testing
Example	"All projects must follow coding standards + CI/CD"	"For this app, use ISTQB testing + peer reviews"	"Run code reviews + automated tests to check compliance"

Techniques to Enhance Software Quality

✅Inspections

Why is it important?

Definition	● **When a piece of work is completed, copies are distributed to co-workers who examine the work, noting defects or problems**

	• A meeting discussing the work and a list of defects requiring rework is produced
Example	• Examine table design (no of fields in a table? too little? too many?) • Program codes (length of variable names?)
Benefits	• Very effective in **removing superficial 表面 errors** (e.g. register, sign up for an account) • **Enhance team spirit** • Helps **spread good programming practices** as the participants discuss specific pieces of code • Helps developers to **produce better structured and self-explanatory software**

Cleanroom Software Development

Definition	• Improved from structured programming: de-component complex system • Each component has only one entry and exit point which make testing easier and more precise

Purpose	• Avoid defects rather than detect and repair them:
	  ○ Use incremental development approach
	  ○ Use statistical testing / verification

Teams	• **Specification team**
	  ○ **Obtains the user requirements** and a usage profile estimating the volume of use for each feature in the system
	• **Development team**
	  ○ **Develops the code** in increments but does **not do testing** on the program code produced
	• **Certification team**
	  ○ **Carries out testing** which is continued until a statistical model shows that the failure intensity has been reduced to an acceptable rate

Benefits	• **Lower number of errors** ○ Codes are develop by development team but testing

	are carried out by certification-team ● **Software of higher quality** ○ Lower no of errors means higher software quality ● **Lower cost** ○ Lower no of errors also means there will be less rework, less rework is a cost saving ● **Project on schedule** ○ Due to less rework
Weaknesses	● Works well with **skilled** and **committed** engineers only ● This approach is confined to 局限于 a **few technologically advanced organizations** only

Quality Circle

Definition	● A **group of 4 to 10 volunteers working in the same area** (e.g. software development department) ● They **meet for about an hour a week to identify, analyze and solve their work-related problems**
Workflow	● One volunteer is the group leader ● There could be an outsider, a facilitator which can advise on procedural matters ● Training is needed to make quality circle work effectively 1. The group **selects a pressing problem that affects their work** 小组选择一个影响其工作的紧迫问题 2. **Identify the cause of the problem** 3. **Decide on a course of action to remove the problems**
Purpose	● **Fix the process so that the same types of error will not occur again** (since faulty process will repeatedly produce faulty products)

Lessons Learnt Reports

Definition	**Written by the project manager** as soon as possible after the completion of the project

Including	• **Reflect on the performance of the recently completed project** when the experience is still fresh • **Identify lessons to be applied to future projects**
Weakness	There is usually **very little follow-up on the recommendation of the reports** because nobody within the organization takes the responsibility and authority to do so

Quality Management & Costs

Costs of Quality

Costs of quality = Cost of achieving quality + Cost of low quality software	
Cost of achieving quality	Example: • Costs incurred in performing quality-related activities
Cost of low quality software	Example: • Rework to correct an error • Resolve complaint • Provide customer support

Categories of Quality Cost

Prevention Costs	Includes the cost of: • Management activities: to **plan and coordinate all QA and QC activities** • Technical activities: to **develop complete requirements and design models** • To **conduct training associated with these activities** Goal: • **Ensure that a project is error-free or within an acceptable range**
Appraisal Costs 评估费用	Associated with **checking processes and its outputs** Includes the cost of:

	• **Conduct technical reviews** • **Test products** • **Collect data and report inspection** Goal: • **Ensure that a project is error-free or within an acceptable range**
Failure Costs	**Internal failure costs** (Costs incurred prior to product delivery): • **Costs to correct design errors** • **Costs to correct program errors** **External failure costs** (Costs incurred after the product has been delivered to customers): • **Resolving customer complaint** • **Handling product return and replacement** • **Providing help line support** • **Managing poor reputation and loss of business**

Quality Assurance & Standards

Quality Standards

Definition	Sets of guidelines, systems, methods, requirements and specifications followed by an organization to ensure consistent process, product and service quality
Importance	• Ensure that products, processes and services meet customer expectations and/or comply with regulatory requirements • Encapsulation of best practice - avoids repetition of past mistakes • They are a framework for defining what quality means in a particular setting • Provide continuity - new staff can understand the organization by understanding the standards that are used

Process Standards

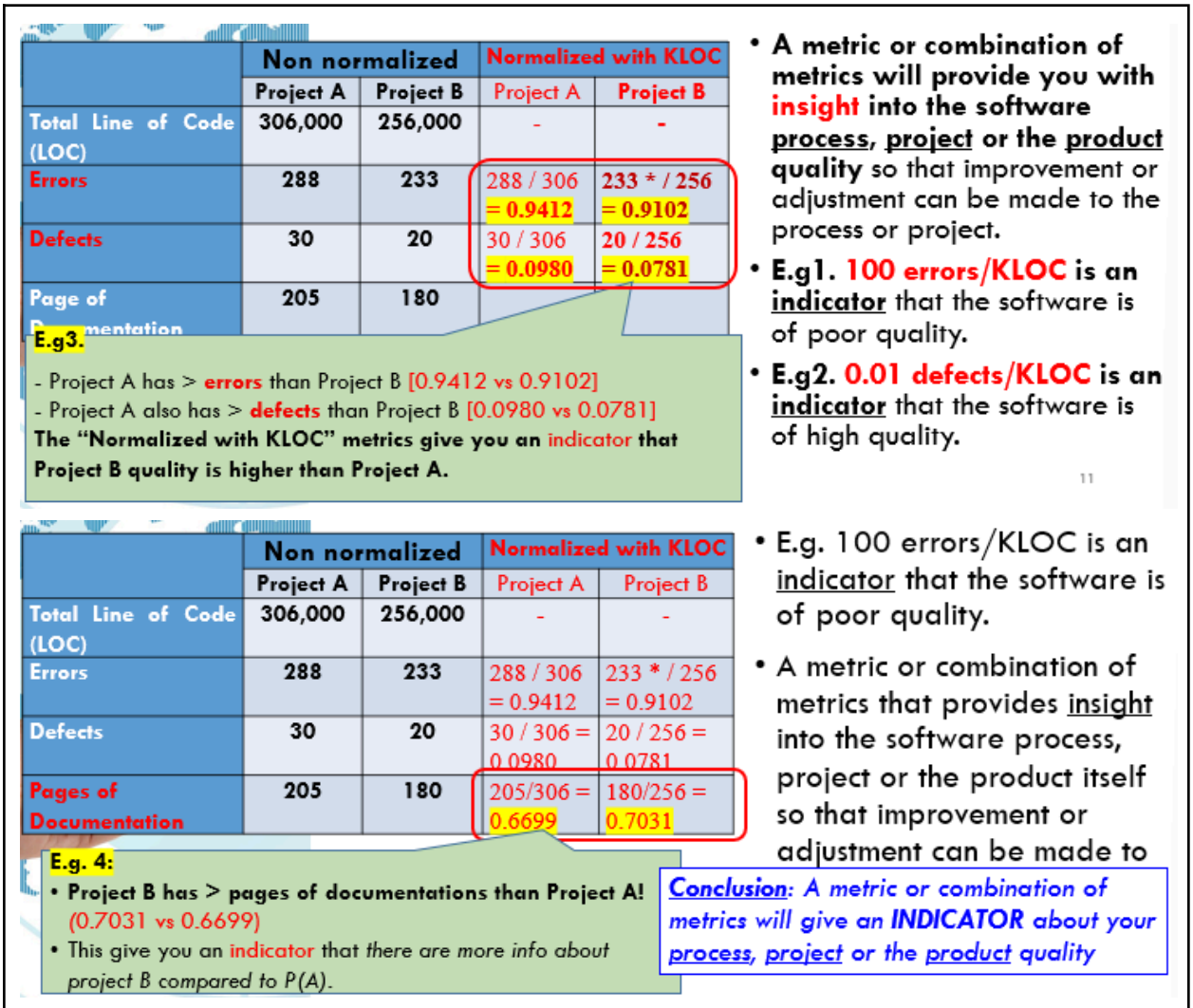
Definition	Define the processes that should be followed during software development
Including	● Definitions of specification ● Design and validation processes ● Process support tools ● Description of documents that should be written during these processes ● Conduct design review ● Submission of new code ● Version release process ● Project plan approval process ● Test recording process ● Change control process

Product Standards

Definition	Apply to the software product being developed including its documentation
Including	● Document standards: ○ Structure of requirements documents ○ Documentation standards ○ Coding standards ○ Design review team ○ Method header format ○ Java programming style ○ Project plan format ○ Change request form

C4: Software Metrics & Measurements

Indicators



	Non normalized		Normalized with KLOC	
	Project A	Project B	Project A	Project B
Total Line of Code (LOC)	306,000	256,000	-	-
Errors	288	233	288 / 306 = 0.9412	233 * / 256 = 0.9102
Defects	30	20	30 / 306 = 0.0980	20 / 256 = 0.0781
Page of Documentation	205	180		

E.g3.

- Project A has > errors than Project B [0.9412 vs 0.9102]
- Project A also has > defects than Project B [0.0980 vs 0.0781]

The "Normalized with KLOC" metrics give you an indicator that Project B quality is higher than Project A.

- A metric or combination of metrics will provide you with insight into the software process, project or the product quality so that improvement or adjustment can be made to the process or project.
- E.g1. 100 errors/KLOC is an indicator that the software is of poor quality.
- E.g2. 0.01 defects/KLOC is an indicator that the software is of high quality.

11

	Non normalized		Normalized with KLOC	
	Project A	Project B	Project A	Project B
Total Line of Code (LOC)	306,000	256,000	-	-
Errors	288	233	288 / 306 = 0.9412	233 * / 256 = 0.9102
Defects	30	20	30 / 306 = 0.0980	20 / 256 = 0.0781
Pages of Documentation	205	180	205/306 = 0.6699	180/256 = 0.7031

E.g. 4:

- Project B has > pages of documentations than Project A! (0.7031 vs 0.6699)
- This give you an indicator that there are more info about project B compared to P(A).

- E.g. 100 errors/KLOC is an indicator that the software is of poor quality.
- A metric or combination of metrics that provides insight into the software process, project or the product itself so that improvement or adjustment can be made to

Conclusion: A metric or combination of metrics will give an INDICATOR about your process, project or the product quality

Normalized Errors	$Normalized\ Errors = \frac{Non\ Normalized\ Errors}{Thousand\ Line\ of\ Code\ (KLOC)}$
Normalized Defects	$Normalized\ Defects = \frac{Non\ Normalized\ Defects}{Thousand\ Line\ of\ Code\ (KLOC)}$
Normalized Pages of Documentation	$Normalized\ Pages\ of\ Documentation = \frac{Non\ Normalized\ Pages\ of\ Documentation}{Thousand\ Line\ of\ Code\ (KLOC)}$

Categories of Software Metrics

Product Metrics

Static Metrics	• Have an indirect relationship with quality attributes. You need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability
	• Measurements made of the system representations such as a design, program or documentation
	• Examples:
	  ○ **Line of code (LOC)** - Measure size of a program, more lines, more complex, more errors
	  ○ **Length of identifiers** (e.g. variable names) - longer, more meaningful
	  ○ **Depth of conditional nesting** (e.g. depth of nested if-statement) - Deeply nested if-statement are hard to understand and potentially error-prone
	  ○ **Cyclomatic complexity** - Measure complexity of a program. More complex means more difficult to test & to maintain

Dynamic Metrics (DM)	• Closely related to software quality attributes. Measurements of a program / system in execution
	• Examples:
	  ○ **Time required to start up an app** (e.g. time req. to start up the WA web)
	  ○ **Execution time required for a particular function** (e.g. time required to generate a class code)
	  ○ **The number of system failure** (e.g. 2021 - 3 failures, 2022 - 2 failures)
	• DM can be **measured / collected during system testing or after system implementation**

Process Metrics

Time	• **The time taken for a particular process to be completed**
	• This can be the total time devoted to the process, calendar time, the time spent on the process by particular engineers, and so on.
	• Examples:
	  ○ Elapsed time - time a request is received until evaluation is complete
	  ○ Time elapsed from completion of evaluation to assignment of change order to personnel
	  ○ Time required to make the change
	  ○ Calendar time expended

Resources	• **The resources required for a particular process**
	• Resources might include total effort in person-days, travel costs or computer resources
	• Examples:
	  ○ Effort (person-hr) to perform the evaluation
	  ○ Effort required to make the change
	  ○ Human effort expended

Events	• **The number of occurrences of a particular event**
	• Examples of events include:
	  ○ 5 runtime errors encountered during testing
	  ○ 2 requirements changes received at the design stage
	  ○ Measures of errors uncovered before the release of the software
	  ○ Defects delivered to and reported by users
	  ○ Errors uncovered during work to make change

Public	• Integrated information that was originally private to individuals and teams
	• Examples:
	  ○ Project level defect rates
	  ○ Effort
	  ○ Calendar times
	  ○ Related data to uncover indicators to improve organizational process performance

Private	• Process metrics that should be private to the individual software engineer and serve as an indicator for the individual only
	• Examples:
	  ○ Defect rates (by individual)
	  ○ Defect rates (by module)
	  ○ Errors found during development

Project Metrics

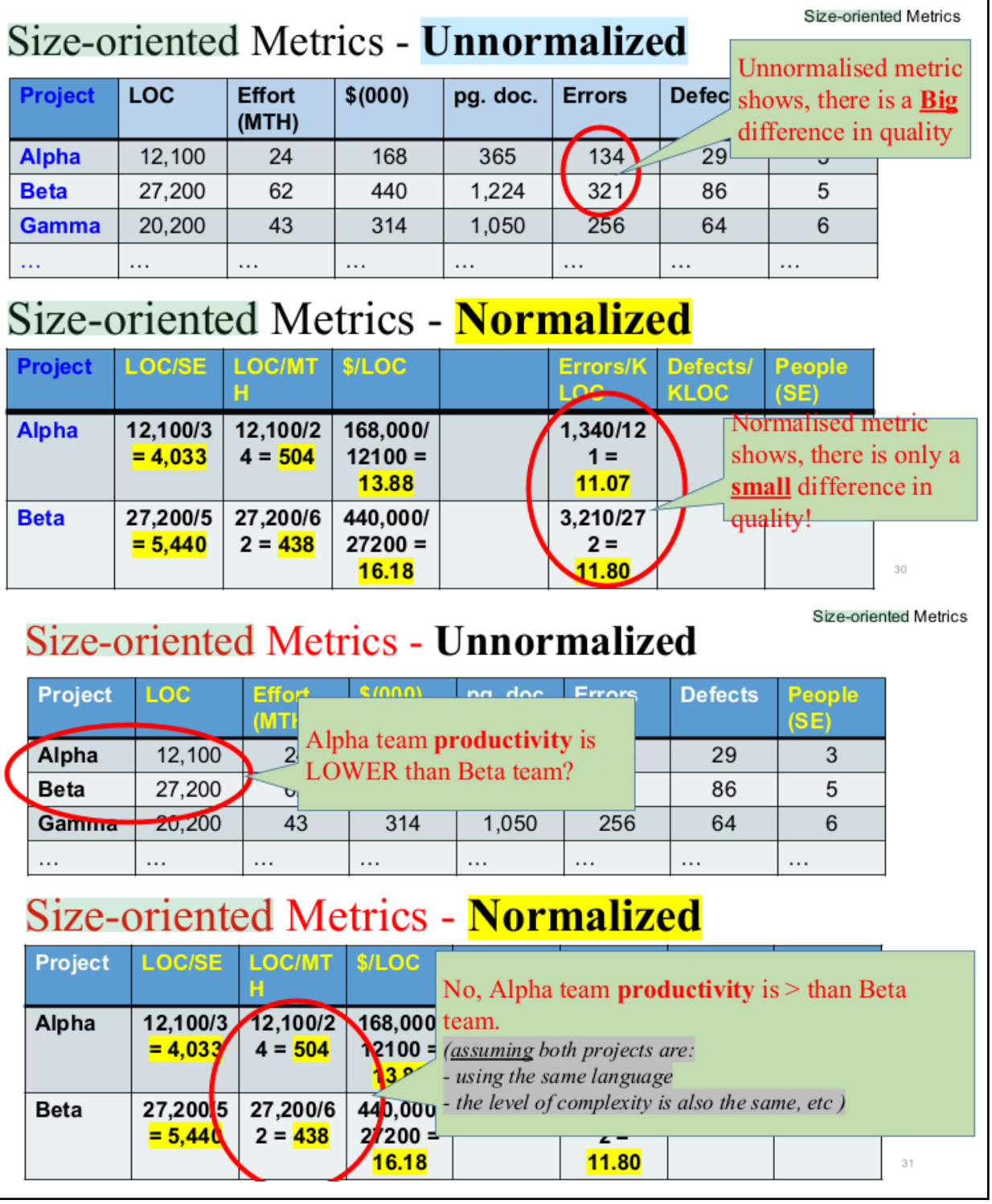
• Used by Project Manager and team to:
  ○ Monitor progress during software development
  ○ Adapt project workflow and technical activities
  ○ Control product quality
• Compare actual metrics with planned metrics to:
  ○ Make necessary adjustments to avoid delays and mitigate potential problems and risks
  ○ Assess product quality on an on-going basis and modify the technical approach to improve quality	

Project Metrics can be collected:	
During estimation	• **Metrics collected from past projects** are used as a basis from which **effort and time estimates are made for current work**
During the execution of current project	• **Compare scheduled dates & actual milestone date to monitor progress** • **Make necessary adjustments** to avoid delays and mitigate potential problems and risks
During the progress of technical work	• Count errors uncovered per review • Distribution of effort per SE task • Pages of documentation per SE task • Delivered source lines per SE task • Function points per SE task

✅Size-Oriented Metrics

- Size-oriented metrics are derived by normalizing (dividing) with a related measures such as:
    - Quality measures over product size
    - Productivity measures over the effort
- Examples of Normalized Size-Oriented Metrics:
    - **Errors per KLOC (Errors / KLOC)**
    - **Defects per KLOC (Defects / KLOC)**
    - **$ per LOC ($ / LOC)**
    - **Pages of documentation per KLOC (Pages of Doc / KLOC)**
    - **Errors per person-month (Errors / person-month)**
    - **LOC per person-month (LOC / person-month)**
    - **$ per page of documentation ($ / Page of doc)**
- Pros:
    - **Easy to use**
        - Simple to apply by counting things like lines of code or function points, making it quick to estimate effort and costs
    - **Early estimation**
        - Helps predict the size of a project during planning phase, allowing for realistic budgets and timelines before coding starts
    - **Easy comparison across projects**
        - Standardized metrics make it easy to compare projects, aiding in benchmarking and cost prediction
    - **Builds historical data**
        - Tracks past project data (like cost per line of code), improving the accuracy of future estimates
    - **Reduces guesswork**
        - Uses concrete measurements instead of guesses, leading to more consistent and reliable results
    - **Great for large projects**
        - Breaks large projects into manageable chunks, simplifying progress tracking and cost management
- Cons:
    - **Not ideal for object-oriented and high-level languages** (e.g. SQL, UML)
    - **Does not account for complexity or efficiency**
    - This measure is **dependent upon programming language**

- It may be very difficult to estimate LOC in early stage of development
- It cannot measure the size of specification as it is defined on code
- It might be very hard for the users to understand the measure



Function-Oriented Metrics

- Measures the functionality delivered by the software system as a normalized value
- Example of normalized function-oriented metrics:
    - **Errors / FP**
    - **Defects / FP**
    - **$ / FP**
    - **Page of documentation / FP**
    - **FP / person-month**
- Pros:
    - **Resource allocation**
        - Function-oriented metrics aid in allocating resources more effectively by providing an estimate of the size and complexity of the software
    - **Decision support**
        - They aid in decision-making processes by providing objective data to support strategic choices and mitigate risks
    - **Continuous improvement**
        - They provide quantifiable data on the size and complexity of the software, thus supporting the continuous efforts in development methodologies, tools and processes
    - **Project planning and control**
        - They support better project planning and control by providing a clear understanding of the scope and complexity of the software projects
    - **Alignment of expectations**
        - They help to measure the performance against the agreed-upon metrics used in contractual agreements between clients and software development providers. This promotes transparency and alignment of expectations.
- Cons:
    - **Subjectivity**

        - Assigning weights to different functional components can be subjective and vary based on the evaluator's interpretation
    - **Complexity of measurement**
        - They require detailed analysis and categorization of software functions, which is time-consuming and resource-intensive, especially for large and complex systems
    - **Dependency on documentation**
        - Accurate measurement of function-oriented metrics depends upon comprehensive and well-documented functional requirements, which may not be always available
    - **Resistance to change**
        - Organizations using traditional metrics find it difficult to adopt function-oriented metrics, especially if it requires significant changes in measurement and reporting practices
    - **Dependency on tool**
        - The effective use of function-oriented metrics depends upon specialized tools for measurement and analysis. This can add complexity and cost to the measurement process.

Function-oriented Metrics

Exercise: Calculate Function Point

Measurement Parameter	Count	Weighting Factor	C * WF
Number of inputs	2	Complex (5)	10
Number of outputs	4	Simple (2)	8
Number of inquiries	7	Average (4)	28
Number of files	3	Average (7)	21
Number of external interfaces	2	Complex (8)	16
			Count Total= 83

Assuming $\Sigma F_i = 60$

FP = Count Total x $[0.65 + 0.01\ \Sigma F_i]$ = 83 x [0.65 + 0.01(60)]
= 83 x [1.25]
= **103.75**

Function-oriented Metrics

Exercise

a. Calculate the function point (FP) for **Project A and B.**

b. **120** pages of documentation is found in Project A, while **190** pages in Project B. Analyse which project has higher degree of **maintainability**.

Measurement Parameter	Weighting factor	Project A's Count	Project B's Count
Number of inputs	Simple(2)	2	5
Number of outputs	Simple(2)	4	12
Number of inquiries	Average(4)	7	9
Number of files	Average(5)	3	7
Number of external interfaces	Complex(6)	2	3
Assuming $\Sigma F_i = 60$			

FP(A) = 67 x [0.65+(0.01 x 60)] = **83.75**

FP(B) = 123 x [0.65+(0.01 x 60)] = **153.75**

- Pages of doc per FP for **P(A)** = 120/83.75=**1.43**
- Pages of doc per FP for **P(B)** = 190/153.75 =**1.24**

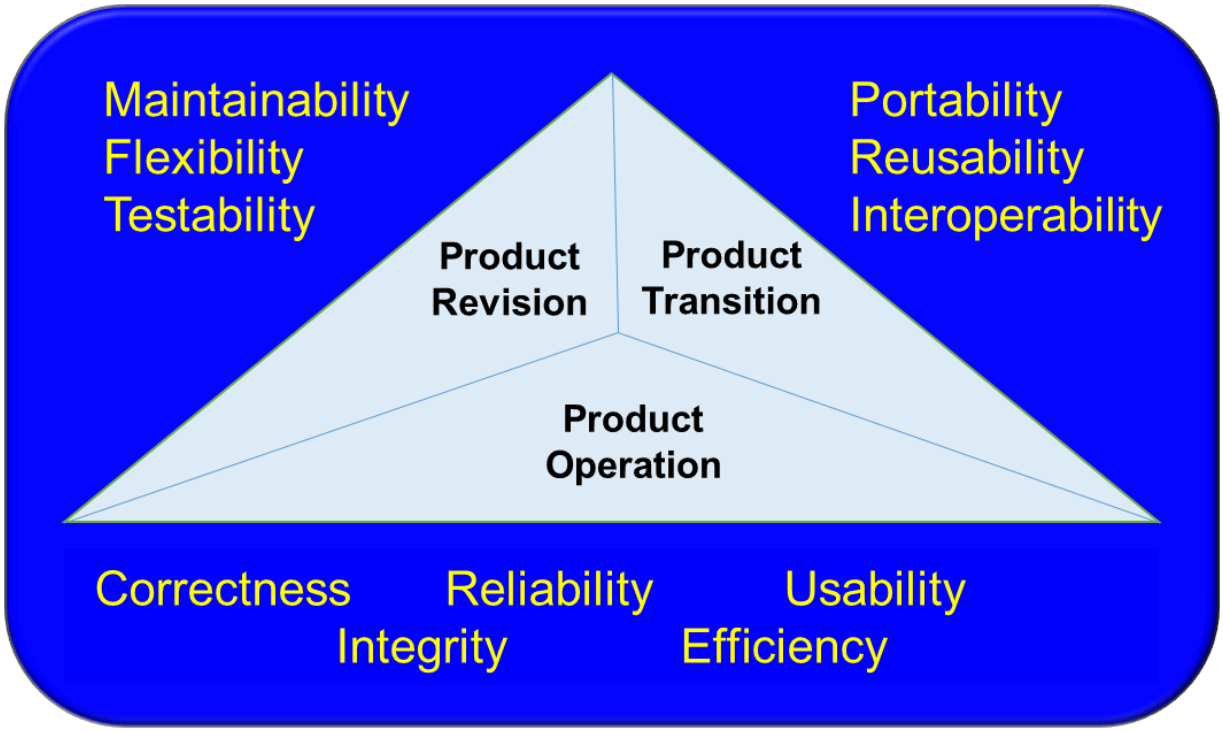
Project A has more Pages of Doc/FP than Project B (1.43 vs 1.24). **Conclusion**: P(A) has higher degree of maintainability than P(B) because there are **more info** abt P(A) than P(B)

$$Function\ Point\ (FP) = Count\ Total \times [0.65 + 0.01\ \Sigma F_i]$$

Count Total	Sum of all FP entries • No. of user inputs • No. of user outputs • No. of user inquiries • No. of files • No. of external interfaces
F_i	Complexity adjustment values where i = 1 to 14

McCall's Software Quality Factors

McCall's Software Quality Factors is a framework used to **evaluate the quality of software products**



Product Operation	• Focus on **how well the software performs during the executions**
	• Key attributes:
	  ○ **Correctness**
	  ■ Does the software meet the requirements?
	  ○ **Reliability**
	  ■ Does it perform consistently without failures?
	  ○ **Efficiency**
	  ■ Does it use resources optimally?
	  ○ **Integrity**
	  ■ Is it secure against unauthorized access?
	  ○ **Usability**
	  ■ Is it easy to use?

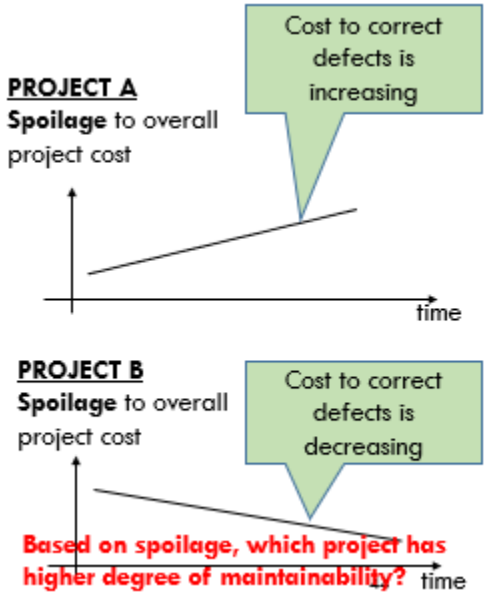
Product Revision	• Deals with **how easily the software can be changed**
	• Key attributes:
	  ○ **Maintainability**
	  ■ How easy is it to fix defects?
	  ○ **Flexibility**

	■ How easily can it adapt to changes? ○ **Testability** ■ How easily can it be tested for errors?
Product Transition	● Focuses on **how well the software can adapt to new environments or integrate with other systems** ● Key attributes: ○ **Portability** ■ Can it run on different platforms? ○ **Reusability** ■ Can components be reused in other projects? ○ **Interoperability** ■ Can it work with other systems?

Metrics for Measuring Software Quality

Usability	● **How easy it is to use the system** ● Can be measured by measuring: ○ **Time required to learn how to perform a task for the first time** (e.g. cropping a video using a video editing software for the first time) ○ **Time required to become moderately efficient in the use of a system** (e.g. time needed to become moderately efficient in operating a car's infotainment system) ○ **User attitude towards the system** (e.g. using a questionnaire then count the no. of negative and positive responses) ○ **Net increase in productivity** (measure the productivity when using the old process or system and measure again the productivity after a user has gained moderate efficiency when using the new process or system)
Maintainability	● **How easy the program can be corrected if an error is encountered**

- **How easy the program can adapt the environment changes**
- **How easy the program can be enhanced if the customer desired a change in requirements**
- No direct way to measure, must use **indirect measures**:
    - **Mean-time-to-change (MTTC)**
        - Time it takes to:
            - analyse the change request
            - design an appropriate modification
            - implement the change
            - testing
            - distribute the change to all users
        - **Highly maintainable programs will have a lower MTTC**
    - **Spoilage**
        - **Cost to correct defects encountered after the software has been released to its end-users**
        - When the ratio of spoilage to overall project cost is plotted as a function of time, a manager can determine whether the overall maintainability of software produced by a software development organization is improving



Correctness	• The degree to which the **software performs its intended functions accurately and without errors**
	• Common measures:
	  ○ **Defects / KLOC**
	  ○ **Defects over a standard period of time** (e.g. 1 year)
	
	Defects/KLOC calculation:
	E.g. If a program has 100 defects and 10,000 LOC, the Defects/KLOC = (100 / 10,000) * 1,000 = 10

Characteristics of Good Metrics

Consistent in its use of units and dimensions	• The mathematical computation of the metric should **use measures that do not lead to bizarre combinations of units** 度量衡的数学计算应使用不会导致奇异单位组合的度量衡 • e.g. Use size and / or FP throughout all projects • e.g. If you measure size in Function Points (FP), stick with FP everywhere. Do not mix FP with LOC in a way that creates strange or meaningless results
Programming language independent	• Metrics should be **based on the analysis model, the design model or the structure of the program itself** • **LOC is programming language dependent** (not suitable for comparing projects that uses different language) • e.g. LOC (Line of Code) is not ideal because 10 lines in Python might equal 50 in Java. Instead, use metrics based on design, analysis or functionality so you can compare projects fairly, no matter the language
Simple and	• Relatively **easy to learn how to derive the**

computable	**metric** • Its computation **should not demand excessive amount of effort or time** • e.g. It should not require huge effort, complex tools or weeks of analysis. If it takes longer to measure than to code, it is not a good metric
Empirically and intuitively persuasive **经验和直觉的说服力**	• **Metrics should match the practitioner's notations about the product attribute under consideration** **衡量标准应与从业人员对所考虑的产品属性的说明相匹配** • Practitioners should feel the metric reflects reality • e.g. "Defects per Function Point" make sense - more functionality usually means more chances for defects
Consistent and objective	• **Always yield results that are unambiguous** **明确无误** • Two people measuring the same project should get the same number • e.g. No room for personal interpretation - it should be fact-based, not opinion-based
An effective mechanism for high-quality feedback	• **Should motivate team for software development improvement** • The metrics should lead to a higher-quality end product • e.g. If a team sees their productivity metric (e.g., FP per person-month) improving, it encourages them to keep refining their process

C5: Risk Management

Category of Risk

Technology to be built (TE)	• Risks associated with the **complexity of the system** to be built and the **"newness" of the technology** that is packaged by the system • Examples: ○ Compatibility issues when completed software needs to integrate with existing legacy systems

Staff size and experience (ST)	• Risks associated with the **overall technical and project experience of the software engineers** who will do the work
	• Examples:
	  ○ Operating with a smaller team than required
	  ○ Team members lack experience in critical areas like UI design and security

Customer characteristics (CU)	• Risks associated with the **sophistication of the customer** and the **developer's ability to communicate with the customer in a timely manner** 与客户的复杂程度和开发商与客户及时沟通的能力有关的风险
	• Examples:
	  ○ Customer has unrealistic expectations
	  ○ Customer keep changing requirements

Product size (PS)	• Risks associated with the **overall size of the software** to be built or modified
Business impact (BU)	• Risks associated with the **constraints imposed by management or the marketplace**
Process definition (PR)	• Risks associated with the **degree to which the software process has been defined and is followed by the development organization**

Category of Risk Strategies

Reactive Risk Strategies (RRS)	• **Take actions to deal with the risks after problems have arisen** • A.k.a. Fire-fighting • e.g. Provide training to team members **after** seeing them struggling with unfamiliar tools
Proactive Risk Strategies (PRS)	• **Identify potential risks, assess the probability and impact before any technical work begins** • **Develop a contingency plan to respond to the risk** in a controlled and effective manner (preparing in advance what to do if any identified risk materialises) • e.g. Provide training to team members on unfamiliar tools **before** project starts

Risk Score

Calculate Risk Score:

Risks	Category	Probability	Impact	Risk score
Less reuse than planned	PS	70%	2	0.7 x 2 = 1.4
End users resist system	BU	40%	3	0.4 X 3 = 1.2
Funding will be lost	CU	40%	1	0.4 X 1 = 0.4
Lack of training tools	DE	80%	3	0.8 x 3 = 2.4
High staff turnover	ST	60%	2	0.6 x 2 = 1.2

Example

Calculate Risk Score:

Risks	Category	Probability	Impact	Risk score
Lack of training tools	DE	80%	3	0.8 x 3 = 2.4
Less reuse than planned	PS	70%	2	0.7 x 2 = 1.4
End users resist system	BU	40%	3	0.4 X 3 = 1.2
High staff turnover	ST	60%	2	0.6 x 2 = 1.2
Funding will be lost	CU	40%	1	0.4 X 1 = 0.4

1. Sort risk table by "**risk score**" in ____ order
2. Set cutoff line

$Risk\ Score = Probability \times Impact$

Risk Exposure (RE)

5.5 Risk Projection – Assessing Risk Exposure (RE)

~~Q. What~~ does this means? Calculate the amount of losses if a particular risk becomes real

If costs are associated with a risk, Halstead's **risk exposure (RE)** metric can be calculated and added to the risk table using:

RE = *P x C*

***P*: probability of occurrence for a risk**

***C*: cost to the project should the risk occur**

EXAMPLE

60 reusable software components were planned. Only 70% (42 components) can be reused, 18 components would have to be developed from scratch. The average component is 100 LOC and the cost for each LOC is $14.00.

Risk identified: Only 70% out of 60 software components can be reused. **The remaining 30% (18 components) have to be custom developed.**

Risk probability: 80%

The **cost** to develop the 18 components = 18 x 100 x 14 = **$25,200.**

Risk exposure: RE = P x C
= 0.80 x 25,200
= **$20,160**

amount of losses if the mentioned risk becomes real

$Risk\ exposure,\ RE = P \times C$

P: probability of occurrence

C: cost to the project should the risk occur

✅Risk Mitigation, Monitoring & Management (RMMM)

Risk mitigation	• To **develop options and actions to enhance opportunities and reduce threats to project objectives**
	• **Reduce the probability of the risk occurring** (make it less likely to happen)
	• **Reduce the impact if the risk does occur** (make the impact less severe)
	• Example:
	  ○ Review and adjust salary packages and benefits to be competitive before the project start, to avoid staff from leaving for better pay elsewhere
	  ○ Assign backup staff member for every critical technology
	  ○ Choose a proven, stable technology stack instead of an untested one to reduce the chance of system crashes
	  ○ Break the project into smaller milestones with buffer time, so delays in one area do not derail 脱轨 the whole project
	  ○ Train developers on secure coding practices to reduce the chance of vulnerabilities being introduced

Risk monitoring & control	• A **project tracking activity** with 3 objectives:
	  ○ **Assess whether predicted risks do, in fact occur**
	  ○ **Ensure that risk avoidance steps defined** for the risk are being properly applied
	  ○ **Carry out the risk management plans as risks occur**
	• Example:
	  ○ Assess the interpersonal relationships among team members

	○ Monitor the availability of jobs within the company and outside ○ Regularly run load tests to see if the system is slowing down under expected traffic ○ Track actual spending vs. planned budget every month to catch overspending early ○ Hold regular check-ins to spot burnout 倦怠, conflicts or declining morale 士气下降 before they escalate
Risk management and contingency planning	● **Take actions when mitigation efforts have failed and that the risk has become a reality** ● Example: ○ Use backup staff that has been planned in the mitigation stage ○ Ask the individual who is leaving to spend the last week to transfer knowledge to remaining team members ○ If the main server goes down, immediately switch to a backup server (disaster recovery plan) ○ If a supplier cannot deliver, use a pre-approved secondary vendor ○ If a lead developer resigns, bring in the backup person trained during mitigation, and reassign tasks to keep progress steady ● Risk management strategies: ○ **Risk avoidance**: Take the necessary actions to avoid the risk altogether ○ **Risk reduction**: Take precautions to reduce the probability of the risk ○ **Risk mitigation**: Take the necessary actions to reduce the impact of the risk when it occurs ○ **Risk transfer**: Transfer the risk to another person or organization ○ **Risk acceptance**: The "do nothing" option. In the risk prioritization process, it would have been decided for some risks to be ignored in order to concentrate on the more likely or

	damaging risks. For some risks the costs of action > the damage inflicted in order to reduce the probability of a risk from happening.

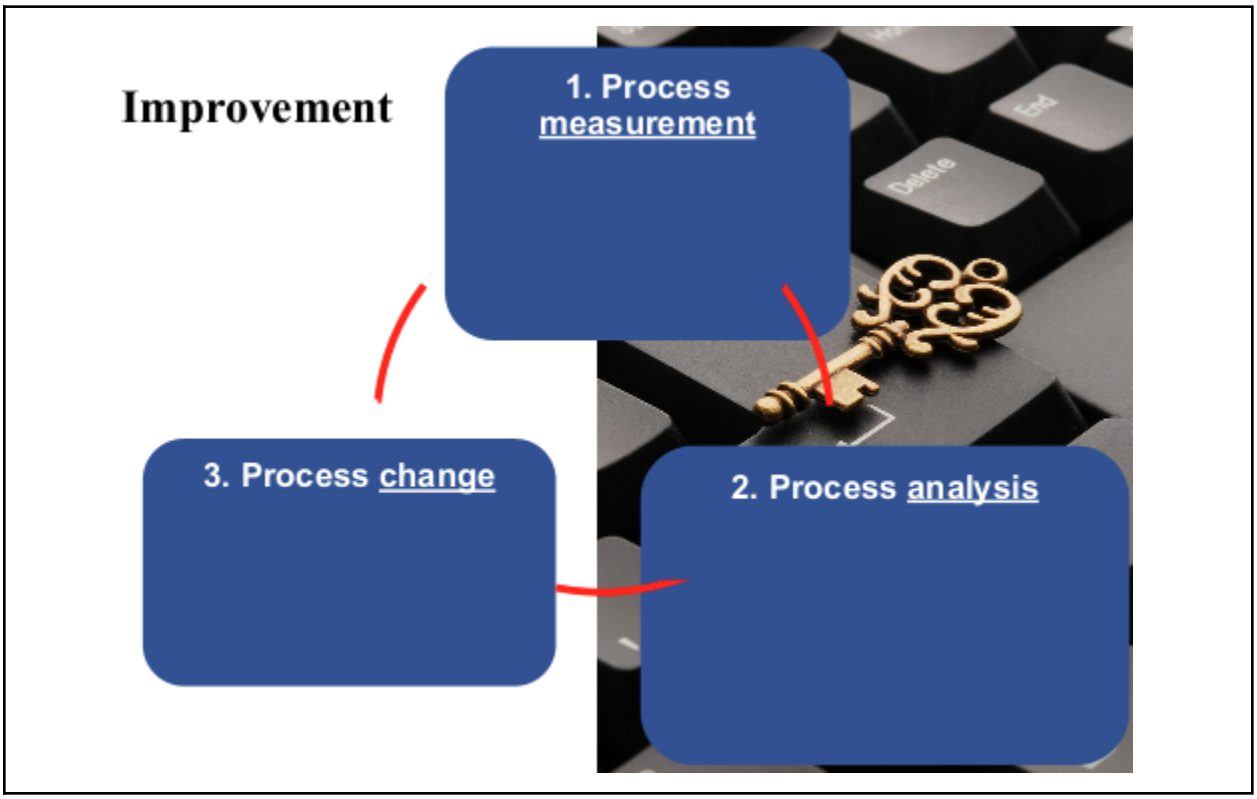
C6: Software Process Improvement

Process Characteristics / Attributes

Understandability	• To **what extent is the process explicitly defined** and **how easy is it to understand the process definition**?
Standardization	• To **what extent is the process based on standard generic processes**? • This may be important for some customers who require conformance with a set of defined process standards. • To **what extent is the same process used in all parts of a company**?
Visibility	• **Do the process activities culminate in clear results**, so that the **progress of the process is externally visible**?
Measurability	• Does the **process** include data collection or other activities that **allow process or product characteristics to be measured**?
Supportability	• To **what extent can software tools be used to support the process activities**?
Acceptability	• Is the **defined process acceptable to and usable by the engineers** responsible for **producing the software product**?
Reliability	• Is the process designed in such a way that **process errors are avoided or trapped before they result in product errors**?
Robustness 稳健性	• Can the **process continue in spite of unexpected problems**?
Maintainability	• Can the **process evolve to reflect changing organizational requirements or identified process improvements**?
Rapidity	• **How fast can the process of delivering a system from**

	a given specification be completed?

Software Process Improvement Cycle



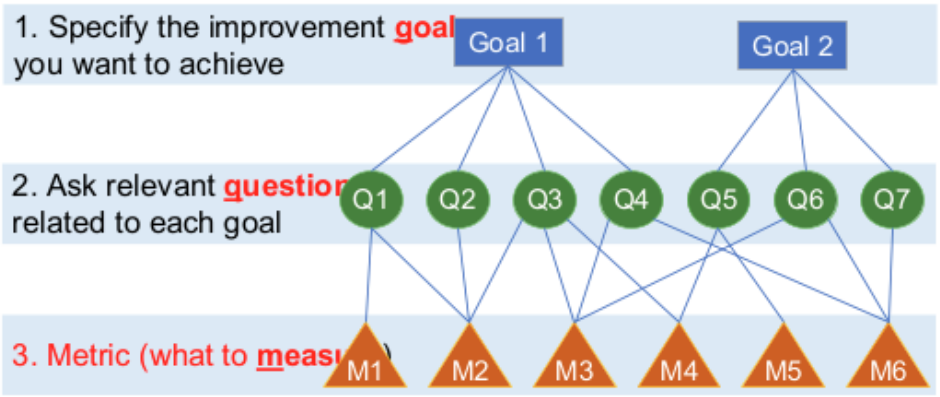
Process Measurement

Definition	**Measure relevant attributes of current process**

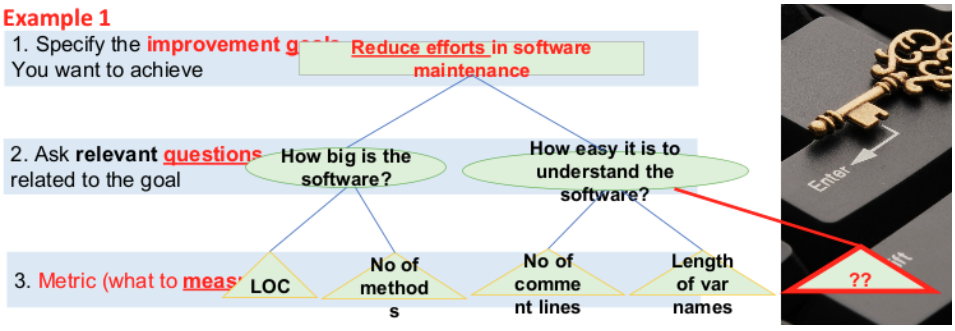
Process Metrics Collection	• Process Metrics Collection:
	  ○ **Time taken for process activities to be completed**
	  ■ Time to complete "requirements gathering"
	  ■ Time to complete UI design
	  ■ Time to complete the software development, process from start to finish
	  ○ **Resources required for processes or activities**
	  ■ Total effort in person-days, hardware and software
	  ○ **Number of occurrences of a particular event**

	■ No. of defects discovered 1 week after delivery ■ No. of run-time errors encountered during 1st UAT
Difficulty in process measurement	What information about the process should be collected to support process improvement? Ans: Use **Goal-Question-Metric (GQM)**
Goal-Question-Metric (GQM)	● Goal-Question-Metric (GQM) Paradigm Components ○ **Goal** ■ **What is the organisation trying to achieve?** ■ The objective of process improvement is to satisfy these goals ■ Focus on how the process affects products or the organization itself ■ Examples: ● Shorter product development time ● Increased product reliability ● Improve process maturity level ○ **Questions** ■ **Questions about areas of uncertainty related to the goals** ■ Examples: ● Where are the bottlenecks in our current process? ● How much time is needed to finalize requirements with customers? ○ **Metrics** ■ **Collected to answer the questions and to confirm whether or not process improvements have achieved the desired goal** ■ Examples: ● Time taken to complete each process activity ● Number of formal communications

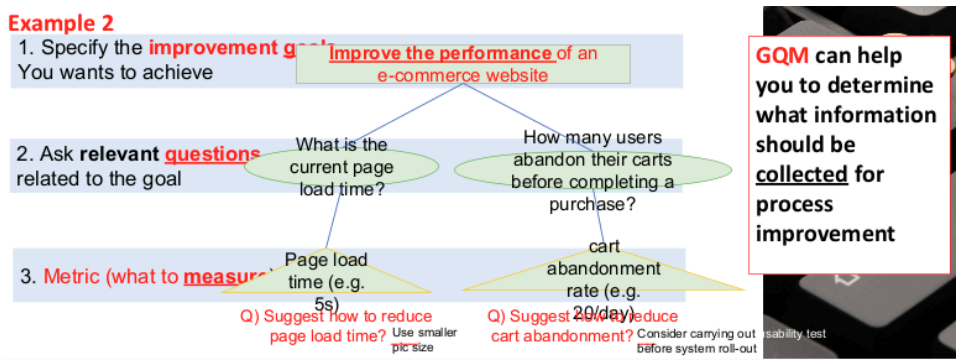
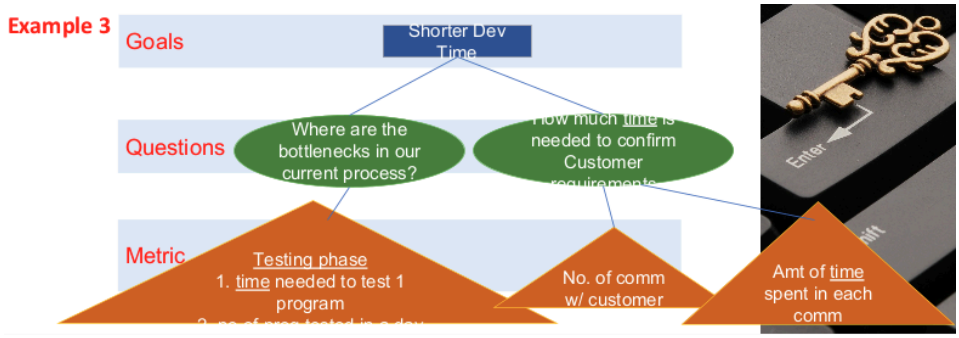
between clients and project manager to confirm requirements
    - Number of defects discovered per test run
- Advantages:
    - Separates organizational concerns (goals) from specific process concerns (questions)
    - **Provides a basis for deciding what data should be collected for …**
    - Suggests collect data should be analyzed in different ways, depending on the question it is intended to answer



Example 1:



Example 2:

	 Example 3: 

Process Analysis

Definition	**Analyse current process to identify bottlenecks and weaknesses**
Objectives	• To **understand the activities carried out in the current process** and the **relationships between these activities** • To **understand the relationships between the process activities** and the **measurements** that have been made • To relate the specific or processes that you are analyzing to comparable processes elsewhere in the organization, or to idealized processes of the same type

Techniques	• **Questionnaires and interviews**
	  ○ Questionnaires
	  ■ Advantage:
	  • Fast / time efficiency
	  • Ease of analysis
	  • Anonymity and confidentiality
	  ■ Disadvantage:

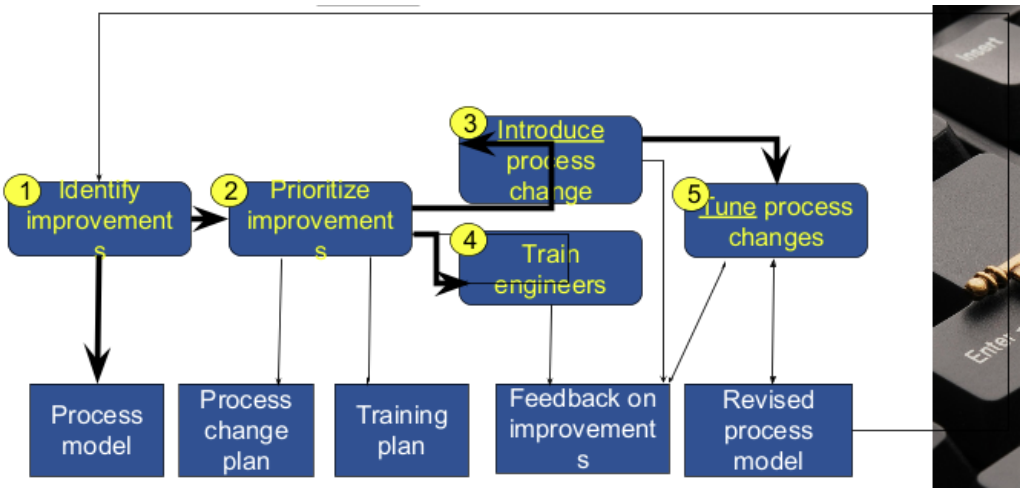
	● If questions are inappropriate, it will lead to incomplete / inaccurate understanding of the process ● Participants (e.g. engineers and PM) treated it as assessment and give answer that they think you want to hear ○ Interviews ■ Advantage: ● Open-ended which can lead to more information ● Immediate feedback ● Flexibility ■ Disadvantage ● Limited sample size ○ Interviews are time-consuming and resource-intensive, making it challenging to conduct them with a large sample size. This limits the generalizability of the findings ● Interviewer effect ○ The skills and demeanor 举止 of the interviewer can impact the interviewee's responses ○ Some interviewees may feel more comfortable with certain interviewers, leading to variations in data ● Time consuming ● **Ethnographic analysis** ○ Involves assimilating 吸收 process knowledge by observation ○ Advantage: ■ Best for in-depth analysis of process fragments rather than for whole-process understanding ○ Disadvantages: ■ Pro-longed process (several months) from initial stages to maintenance stages ■ Impractical for large projects, it may even require

	several years to carry out this analysis ● **Analyse published process models** ○ Process models are suitable for focusing attention on the activities in a process and the information transfer between these activities ○ Process models is optionally formal or complete ○ Purpose is to provoke discussion rather than document the process in detail 目的是引发讨论，而不是详细记录过程 ○ Advantage: ■ Model-oriented questions can be used to help understand the process, e.g.: ● What activities take place in practice but are not shown in the model? ● Are there process activities, shown in the model, that you think are inefficient / problematic? ● If something goes wrong, do you follow the model or take up emergency actions? Why or why not? ● How's the communication? Who is involved? Where are the bottlenecks? ● What tools are used to support the activities in the model? Effective?

Process Change

Definition	● **Change the process to reduce / eliminate bottlenecks and weaknesses** ● Process change must **not be dependent on individuals** ● **Change should be officially enforced by the organization** (make the change permanent by making it a standard practice in company with company-wide support and training)
Including	● **Introducing new practices, methods or processes** ○ Examples: ■ Conduct peer review ● **Change the order of process activities**

	• **Introduce new or remove existing deliverables**
	• **Introduce new roles / responsibilities**
	  ○ Examples:
	  ■ Form quality circle team (then meet up weekly to solve work related problems)
	  ■ Write lesson learnt report

Stages	 1. Improvement identification 2. Improvement prioritization 3. Process change introduction 4. Process change training 5. Change tuning

Difficulties	• **Resistance to change**
	  ○ Project Manager resist process change
	  ■ **Any innovation has unknown risks associated with it**
	  ■ May prefer inefficient but predictable process than an improved process that has organizational benefits but which has short-term risks associated with it
	  ○ Engineers resist resist process change
	  ■ **Resists the introduction of new processes** for similar reasons or because they see **new processes as threatening their professionalism**
	  ■ Feel the new process **gives them less discretion**

	and does not recognize the value of their skills and experience ● **Change Persistence** ○ Changes may be **proposed by an 'evangelist**' who **believes strongly that the changes will lead to improvement** ○ He or she may **work hard to ensure the changes are effective and the new process is accepted** ○ If the **'evangelist' leaves the company**, the people involved may **simply revert to the previous ways of doing things**

C7: Software Configuration Management

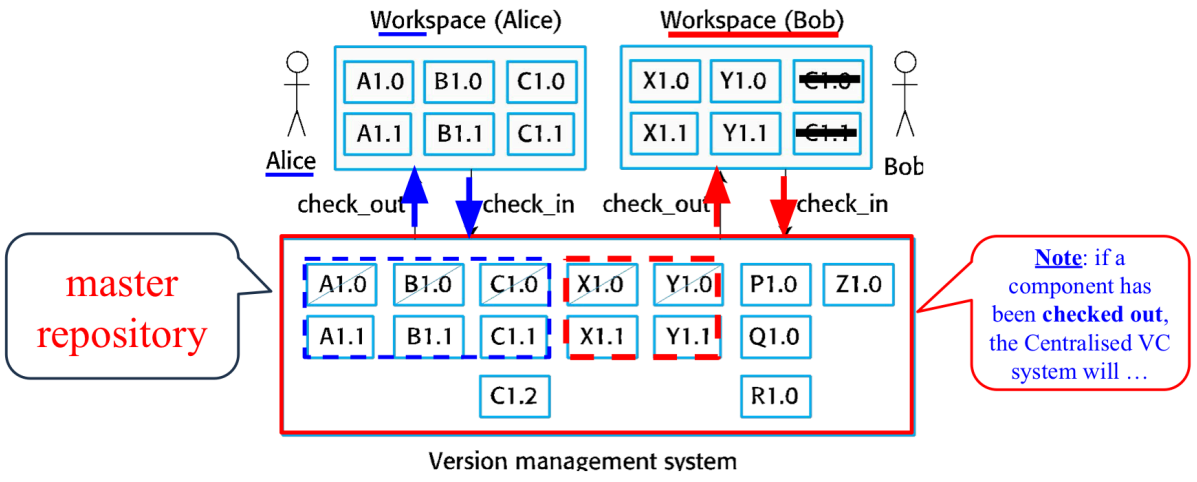
Version Control Systems

Centralized Version Control Systems

Definition	• There is a **single master repository that maintains all versions of the software components** that are being developed • **Subversion** is a widely used example of a centralized VC system
More Info	• If several people are working on a component at the same time, each check it out from the repository. • If a component has been checked out, the VC system warns other users wanting to check out that component that it has been checked out by someone else

Centralized Version Control *(An example of a widely used CVC is **Subversion**)*

7.2 Version Management



19

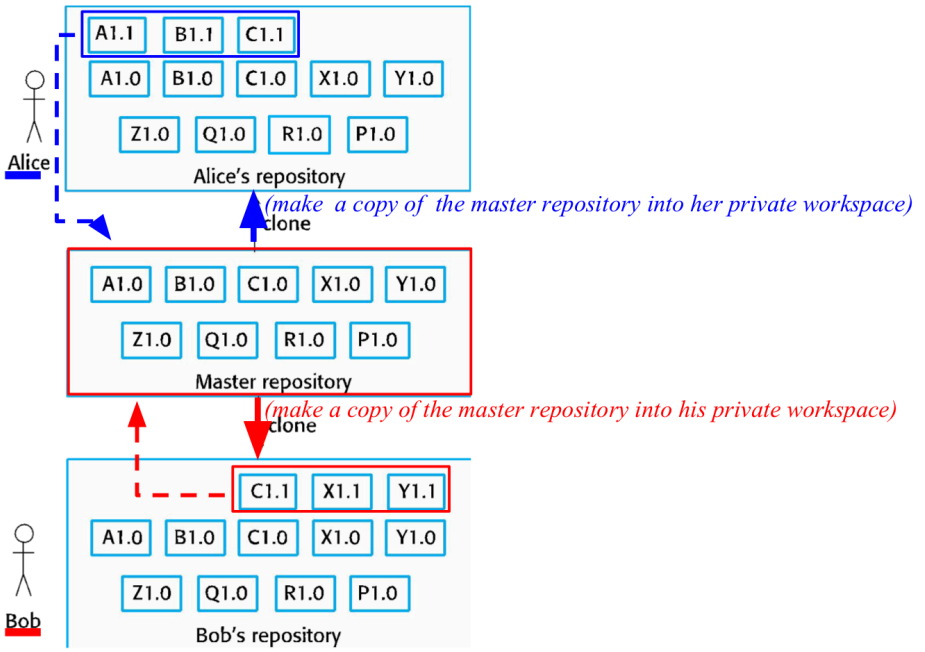
Distributed Version Control Systems

Definition	• **Multiple versions of the component repository exist at the same time** • **Git** is a widely-used example of a distributed VC system
More Info	• A 'master' repository is created on a server that maintains the code produced by the development team • Instead of checking out the files that they need, a developer creates a clone of the project repository that is downloaded and installed on their computer • Developers work on the files required and maintain the new versions on their private repository on their own computer • When changes are done, they 'commit' these changes and update their private server repository. They may then 'push' these changes to the project repository

Benefits	• **Provides backup mechanism for the repository**
	  ○ If the repository is corrupted, work can continue and the project repository can be restored from local copies
	• **Allows for off-line working** so that developers can commit changes if they do not have a network connection
	• **Project support** is the default way of working
	  ○ Developers can compile and test the entire system on their local machines and test the changes that they have made

Distributed Version Control

*(An example of a widely used DVC is **Git**)*



21

Differences between Centralized and Distributed Version Control Systems

	Centralized Version Control System (CVCS)	**Distributed Version Control System (DVCS)**
Definition	• A single master repository stores all versions of the software components • Developers check out files from this central repository and check them back in after making changes • e.g. Subversion (SVN)	• Every developer has a full copy (clone) of the repository, including its entire history • Developers work locally and later synchronize with others • e.g. Git
How it works	• One central server holds the codebase • Developers connect to this server to get the latest version and to commit changes	• Developers clone the repository to their own machines • Work is done locally (commits, branching, merging)

	• Requires network access to the central server	• Changes are pushed / pulled to a shared remote repository when needed
Pros	• Simple to understand and manage • Easier to enforce access control and security (since everything is in one place) • Good for small teams with stable internet access	• No single point of failure (every clone is a backup) • Supports offline work (commits can be made without internet) • Faster operations (since most actions are local) • Encourages branching and merging, which supports parallel development • Essential for open-source and large distributed teams
Cons	• Single point of failure (If the server goes down, no one can commit or retrieve updates) • Requires constant network connection • Slower collaboration for large, distributed teams	• More complex to learn and manage compared to centralized systems • Requires more local storage (since each developer has a full copy) • Synchronization can be tricky if many developers push conflicting changes)

Factors Influencing System Release

Competition	• For mass-market software, **when a competing product has introduced new features, market share may be lost if a new system release is not provided** to your existing customers
Platform changes	• When a **new version of the OS platform is released**, you may have to **create a new release of the software application for the updated OS**

	• e.g. Microsoft upgraded Windows 10 to Windows 11. Your game software also must be updated to make it compatible with Windows 11. Else old game software will not function smoothly
Technical quality of the system	• When there are **serious system faults which affect many customers**, it may be necessary to **issue a fault repair release** • **Minor system faults** may be repaired by **issuing patches (usually distributed over the internet)** that can be applied to the current release of the system
Marketing requirements	• The marketing department of an organization have made a **commitment for releases to be available at a particular date**

Factors in Change Analysis

Factor	Explanation	Why It Matters
Benefits of the change	Positive outcomes expected from implementing the change	Justifies the change and helps gain stakeholder support
Consequences of not making the change	Risks or problems that may arise if the change is not made	Highlights urgency and the need for action
Costs of making the change	Resources required to implement the change (money, time, effort)	Assesses feasibility and helps with budgeting and planning
Number of users affected	The scale of impact across the user base	Determines the level of communication, support and training needed
Product release cycle	Timing and frequency of product updates or releases	Aligns the change with development schedules to avoid disruption

C8: System Dependability & Critical Systems

Critical Systems

Definition	• A system whose failures can result in:
	  ◦ Significant economic losses
	  ◦ Physical damage
	  ◦ Threats to human life

Type of Critical Systems	• **Safety-Critical Systems**
	  ◦ A system failure may cause:
	  ▪ Injury
	  ▪ Loss of life
	  ▪ Major environmental damage
	  ◦ Examples:
	  ▪ Self-driving cars (failure of the braking system may cause ...)
	  ▪ Emergency Lighting System (system failure could lead to injuries during an evacuation)
	  ▪ Flight auto-pilot control system
	  ▪ Nuclear reactor management system
	• **Mission-Critical Systems**
	  ◦ A system failure may result in the failure of some goal-directed activity
	  ◦ Examples:
	  ▪ GPS (Using GPS, the goal is ...) (If GPS fails, ...)
	  ▪ Attendance system (The goal of attendance system is ...) (If the attendance system failed, ...)
	  ▪ Campus car plate recognition system
	• **Business-Critical Systems**
	  ◦ System failure can have serious consequences for the business
	  ◦ Examples:
	  ▪ ERP System (ERP system failure can result in serious consequences for the business)
	  ▪ Online banking system (OBS failure also can have serious consequences for the business)
	  ▪ E-commerce website

Causes of System Failure	• **Failure of hardware**
	  ○ Due to design and manufacturing errors or components have reached their end of life
	• **Failure of software**
	  ○ Due to errors in code
	• **Failure of operation**
	  ○ Due to human errors, perhaps the largest single cause of system failures in socio-technical systems 人为失误可能是造成社会技术系统故障的最大单一原因
	  ○ Examples:
	  ■ Accidentally deleted some important system files such as .ini files, .exe files, .sys files, etc
	  ■ Not configuring a system correctly

Cost of System Failure	• **Direct failure costs**
	  ○ System need to be replaced
	• **Indirect failure costs**
	  ○ Lost of revenue & reputation
	  ○ Litigation (lawsuits) costs 诉讼（官司）费用
	
	For critical systems, the cost of failure is often very high.

✅Dependability

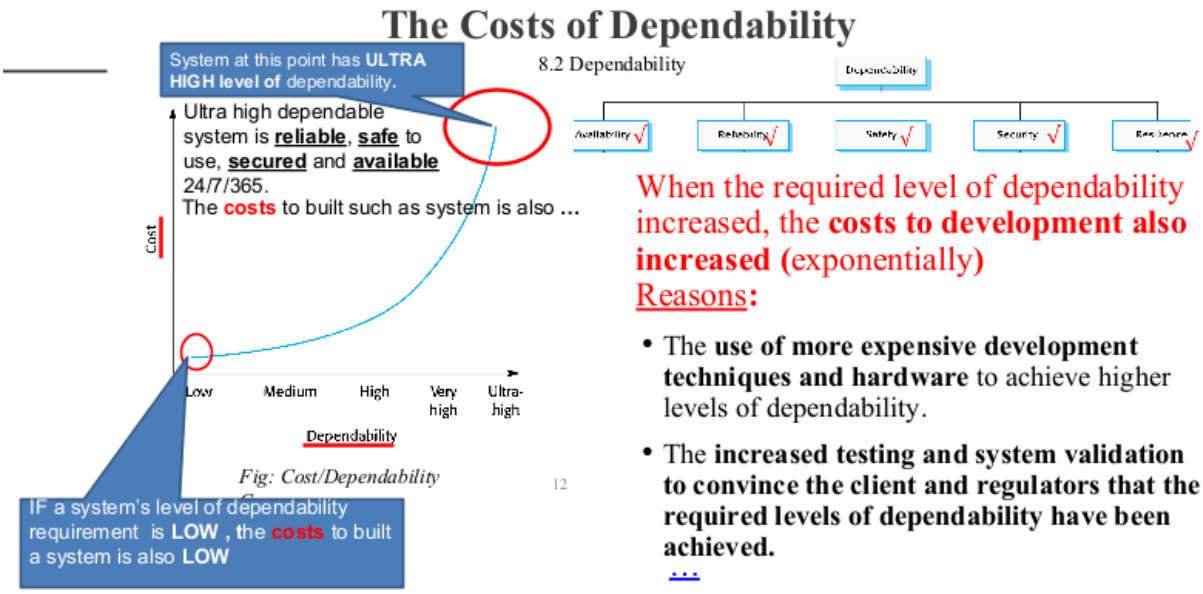
Definition	• How **reliable** and **consistent** a system is • A dependable system is one that will **work as expected, without any problems or failures** • A dependable system will **operate as expected** and that it will **not fail in normal use**

✅Dimensions of Dependability (5)

Availability	• The ability of a system to be **up and running** and able to **deliver useful services to users when needed** OR • The probability that the system will be **up and running** and **able to deliver useful services to users**

Reliability	• The ability of a system to **correctly deliver services as expected by users over a period of time** OR • The probability that the system will **correctly deliver services as expected by users**
Safety	• The ability of a system to **prevent or mitigate accidents, injuries, and other harmful events** OR • A judgment of **how likely it is that the system will cause damage to people or its environment**
Security	• The ability of a system to **resist accidental or deliberate attacks** 蓄意攻击 OR • A judgment of **how likely it is that the system can resist accidental or deliberate attacks**
Resilience 复原力	• The ability of a system to **maintain continuity of its critical services in the presence of disruptive events** 破坏性事件 such as equipment failure and cyber-attacks, etc OR • A judgment of **how well a system can maintain the continuity of its critical services in the presence of disruptive events** such as equipment failure and cyber-attacks
Others	
Repairability	Reflects the extent to which the system can be repaired in the event of a failure
Maintainability	Reflects the extent to which the system can be adapted to new requirements
Error tolerance	Reflects the extent to which user input errors can be avoided and tolerated

Cost / Dependability Curve



- The **higher** the **level of dependability**, the **higher** the **cost of development**.
- Reasons:
    - **Implementation of more expensive development techniques and hardware** to achieve higher dependability level
    - **Increasing of testing and system validation** to convince the client and regulators that the required levels of dependability have been achieved

Approaches to Achieve Dependability

Redundancy	• **Keep more than one version of critical components so that a backup is available** • Example: ○ If availability is critical (e.g. e-commerce systems), companies keep backup servers and switch to backup servers automatically if failure occurs
Diversity	• **Provide the same functionality in different ways in different components so that they will not fail in the**

	same way ● Example: ○ To provide resilience against external attacks, have different servers using different operating systems (e.g. Windows Server & Linux)
● Redundant and diverse components should be independent to avoid "common-mode" failures ● e.g. Components implemented in different programming languages means that a compiler fault will not affect all of them	

Fault Management (Approaches to Improve System Reliability)

Fault avoidance	● **Develop techniques to either minimize the possibilities of mistakes and / or trap mistakes before these result in system failure** ● e.g. **Avoid error-prone programming language constructs** like pointers, use static analysis (check program codes to find unusual or problematic patterns) to detect program anomalies
Fault detection	● Use **verification** (check whether the software development process is carried out correctly) and **validation** (check whether developed system is working as expected) techniques to **discover and remove faults** in a system before it is deployed ● e.g. implement systematic system testing and debugging
Fault tolerance	● **Incorporate self-checking facilities in a system** 在系统中加入自我检查设施 ○ e.g. Check available memory and disk space ● **Use of redundant system** ○ e.g. Run a secondary system at the same time with the primary system

✅Factor Affecting Availability Perception

The number of	● Loss of service in the middle of the night is less

users affected by the service outage	important for many systems than loss of service during peak usage periods ● Examples: ○ If a banking app goes down at 3 AM, only a few night owls notice ○ If it goes down at 12 PM on payday, thousands of users are frustrated ● Timing + scale of impact matter more than just the outage itself
Duration of system unavailability	● The longer the outage, the more the disruption ● Several short outages are less likely to be disruptive than one long outage ● Examples: ○ A CI/CD pipeline (build server) going down for 5 minutes may be tolerable, developers can retry builds ○ If the pipeline is down for 3 hours, the entire development team is blocked, delaying releases ● One long outage feels worse than many short ones
* Availability affect Reliability (if a system is unavailable then it is unreliable)	

Safety

Definition	Ability of a system to operate normally or abnormally, without causing human injury or death and without causing damage to the system's environment
Safety vs Reliability	● **Reliability** is concerned with **conformance to a given specification** and **delivery of service** ● **Safety** is concerned with **ensuring system cannot cause damage** irrespective of whether or not it conforms to its specification ● A **reliable system cannot guarantee safety** ● Example: ○ A technician pressed a button that instructed the aircraft utility management software to raise the undercarriage.

	The software carried out the technician's instruction perfectly. If the system can perform the action upon pressing the button without further control, it can be unsafe although it has followed the specification. The system should only allow the command when the plane is in the air (further control), this can promise safety.

Safety Critical Systems

Definition	• Systems where it is essential that system operation is always safe • e.g. The system should never cause damage to people or the system's environment
Examples	• Control and monitoring systems in aircraft • Process control systems in chemical manufacturing plant • Automobile control systems such as braking and engine management systems
Primary safety-critical systems	• **Embedded software systems whose failure can cause the associated hardware to fail and directly threaten people** • e.g. The insulin pump control system

Secondary safety-critical systems	• **Systems whose failure results in faults in other systems, which can then have safety consequences**
	• Examples:
	  ○ The Patient Information System is safety-critical as failure may lead to inappropriate treatment being prescribed
	  ○ Infrastructure control systems are also secondary safety-critical systems

Security

Security Improvements

Avoid	• Design system so that vulnerabilities do not occur

vulnerability	• e.g. Do not connect system to external public network then there is no possibility of an attack from members of the public (make a choice between convenience and security)
Implement attack detection and elimination	• Design system with the ability to detect vulnerability and remove them before they result in an exposure • e.g. Use virus checkers to find and remove viruses before they infect a system
Limit exposure	• Design system so that successful attack are minimised • e.g. Restore damage IS with backup (company must have a backup policy)