Chapter 22 – Project Management

SWEPM Unit 4

Topics covered

- ✧ Risk management
- ✧ Managing people
- ✧ Teamwork

Project

- ✧ Project = *Temporary Effort* to achieve a specific goal
- ✧ Three characteristics
  - Fixed beginning
  - Fixed ending
  - Fixed scope
- ✧ A typical semester of BE Course – Yes
- ✧ Running SIT to provide education – No (Operation)
- ✧ Running SIT from Jan 2023 to Dec 2023 – Yes
- ✧ Running a grocery shop to earn a living – No
- ✧ On time, within budget, high quality, no accidents

Scope

- ✧ Management term
- ✧ Used PM as well as SWE
- ✧ Scope = Description of things to be done
  - List of features of the mobile app
  - List of courses to be cleared to earn BE degree
- ✧ In Scope = Should be done
- ✧ Out of Scope = Should not be done

Program

- ✧ Build a campus for a new engineering college
  - Scope: Entrance gate, walking pathway, admin building, CSE dept, Civil Dept, Workshop, Library, Mess, Hostel-1, Hostel-2
  - Each element in scope = project
  - Building the entire campus = Program
- ✧ Program = Collection of Projects

Portfolio

- What if Running the college + building a new Startup Office
- Running a college = Operation
- Building a new startup office = Project
- Our example = Operation + Project
- It is called *Portfolio*.
- Ex: Running and extending Kempegowda International Airport
  - Running the airport + building new run way + building new terminal
- Namma Metro

Integration	• Coordinate activities across all project management areas and process groups
Scope	• Ensure the project work includes all elements required to complete the work
Schedule	• Ensure the project work is completed in a timely way
Cost	• Plan, estimate, manage and control project finances
Quality	• Ensure the project delivers a quality output that is fit for purpose
Resource	• Secure, manage and monitor use of resources throughout the project
Communications	• Ensure communications on the project are planned and carried out appropriately
Risk	• Identify, assess and manage risk
Procurement	• Carry out purchasing and contracting as required
Stakeholder	• Identify and engage stakeholders throughout the project

Software project management

- Concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

Success criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer's expectations.
- Maintain a coherent and well-functioning development team.

Software management distinctions

- The product is intangible.
  - Software cannot be seen or touched. Software project managers cannot see progress by simply looking at the artefact that is being constructed.
- Many software projects are 'one-off' projects.
  - Large software projects are usually different in some ways from previous projects. Even managers who have lots of previous experience may find it difficult to anticipate problems.
- Software processes are variable and organization specific.
  - We still cannot reliably predict when a particular software process is likely to lead to development problems.

Universal management activities

- *Project planning*
  - Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
  - Covered in Chapter 23.
- *Risk management*
  - Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.
- *People management*
  - Project managers have to choose people for their team and establish ways of working that leads to effective team performance.

Management activities

- *Reporting*
  - Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.
- *Proposal writing*
  - The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.

Risk management

Risk management

- Risk management is concerned with identifying risks and drawing up plans to minimise their effect on a project.
- Software risk management is important because of the inherent uncertainties in software development.
  - These uncertainties stem from loosely defined requirements, requirements changes due to changes in customer needs, difficulties in estimating the time and resources required for software development, and differences in individual skills.
- You have to anticipate risks, understand the impact of these risks on the project, the product and the business, and take steps to avoid these risks.

Risk classification

- There are two dimensions of risk classification
  - The type of risk (technical, organizational, ..)
  - what is affected by the risk:
- *Project risks* affect schedule or resources;
- *Product risks* affect the quality or performance of the software being developed;
- *Business risks* affect the organisation developing or procuring the software.

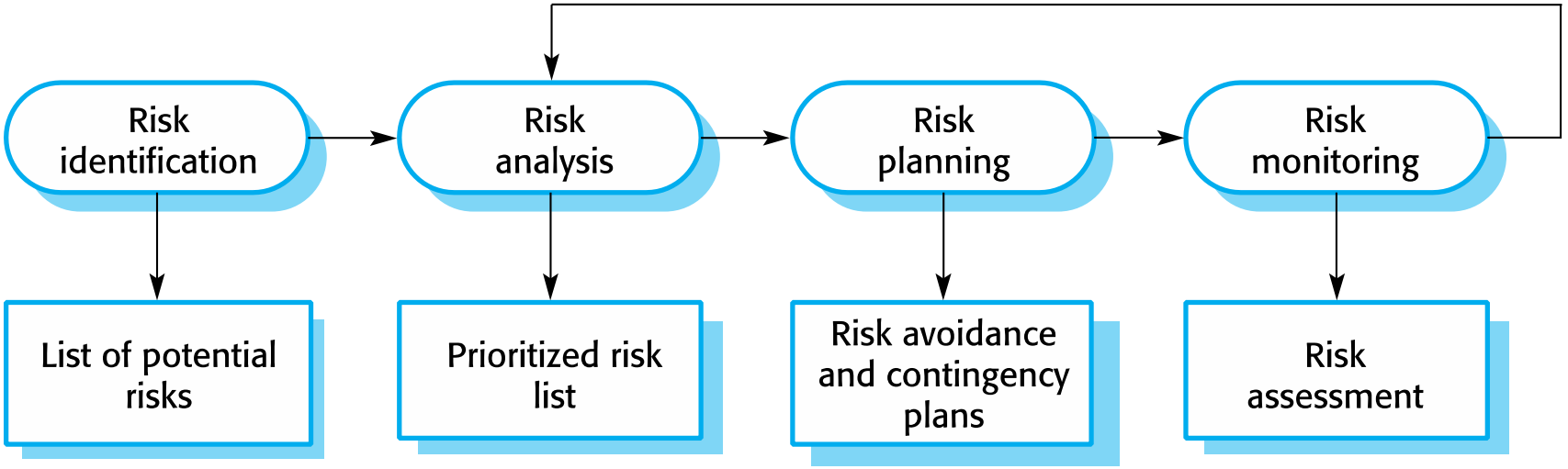
Examples of project, product, and business risks

Risk	Affects	Description
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organizational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule.
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool underperformance	Product	CASE tools, which support the project, do not perform as anticipated.
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

The risk management process

- ✧ Risk identification
  - Identify project, product and business risks;
- ✧ Risk analysis
  - Assess the likelihood and consequences of these risks;
- ✧ Risk planning
  - Draw up plans to avoid or minimise the effects of the risk;
- ✧ Risk monitoring
  - Monitor the risks throughout the project;

The risk management process



Risk identification

- ✧ May be a team activities or based on the individual project manager's experience.
- ✧ A checklist of common risks may be used to identify risks in a project
  - Technology risks.
  - Organizational risks.
  - People risks.
  - Requirements risks.
  - Estimation risks.

Examples of different risk types

Risk type	Possible risks
Estimation	The time required to develop the software is underestimated. (12) The rate of defect repair is underestimated. (13) The size of the software is underestimated. (14)
Organizational	The organization is restructured so that different management are responsible for the project. (6) Organizational financial problems force reductions in the project budget. (7)
People	It is impossible to recruit staff with the skills required. (3) Key staff are ill and unavailable at critical times. (4) Required training for staff is not available. (5)
Requirements	Changes to requirements that require major design rework are proposed. (10) Customers fail to understand the impact of requirements changes. (11)
Technology	The database used in the system cannot process as many transactions per second as expected. (1) Reusable software components contain defects that mean they cannot be reused as planned. (2)
Tools	The code generated by software code generation tools is inefficient. (8) Software tools cannot work together in an integrated way. (9)

Risk analysis

- Assess probability and seriousness of each risk.
- Probability may be very low, low, moderate, high or very high.
- Risk consequences might be catastrophic, serious, tolerable or insignificant.

Risk types and examples

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget (7).	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project (3).	High	Catastrophic
Key staff are ill at critical times in the project (4).	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused. (2).	Moderate	Serious
Changes to requirements that require major design rework are proposed (10).	Moderate	Serious
The organization is restructured so that different management are responsible for the project (6).	High	Serious
The database used in the system cannot process as many transactions per second as expected (1).	Moderate	Serious

Risk types and examples

Risk	Probability	Effects
The time required to develop the software is underestimated (12).	High	Serious
Software tools cannot be integrated (9).	High	Tolerable
Customers fail to understand the impact of requirements changes (11).	Moderate	Tolerable
Required training for staff is not available (5).	Moderate	Tolerable
The rate of defect repair is underestimated (13).	Moderate	Tolerable
The size of the software is underestimated (14).	High	Tolerable
Code generated by code generation tools is inefficient (8).	Moderate	Insignificant

Risk planning

- Consider each risk and develop a strategy to manage that risk.
- Avoidance strategies
  - The probability that the risk will arise is reduced;
- Minimization strategies
  - The impact of the risk on the project or product will be reduced;
- Contingency plans
  - If the risk arises, contingency plans are plans to deal with that risk;

What-if questions

- What if several engineers are ill at the same time?
- What if an economic downturn leads to budget cuts of 20% for the project?
- What if the performance of open-source software is inadequate and the only expert on that open source software leaves?
- What if the company that supplies and maintains software components goes out of business?
- What if the customer fails to deliver the revised requirements as predicted?

Strategies to help manage risk

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

Strategies to help manage risk

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Risk monitoring

- Assess each identified risks regularly to decide whether or not it is becoming less or more probable.
- Also assess whether the effects of the risk have changed.
- Each key risk should be discussed at management progress meetings.

Risk indicators

Risk type	Potential indicators
Estimation	Failure to meet agreed schedule; failure to clear reported defects.
Organizational	Organizational gossip; lack of action by senior management.
People	Poor staff morale; poor relationships amongst team members; high staff turnover.
Requirements	Many requirements change requests; customer complaints.
Technology	Late delivery of hardware or support software; many reported technology problems.
Tools	Reluctance by team members to use tools; complaints about CASE tools; demands for higher-powered workstations.

Managing people

Managing people

- People are an organisation's most important assets.
- The tasks of a manager are essentially people-oriented. Unless there is some understanding of people, management will be unsuccessful.
- Poor people management is an important contributor to project failure.

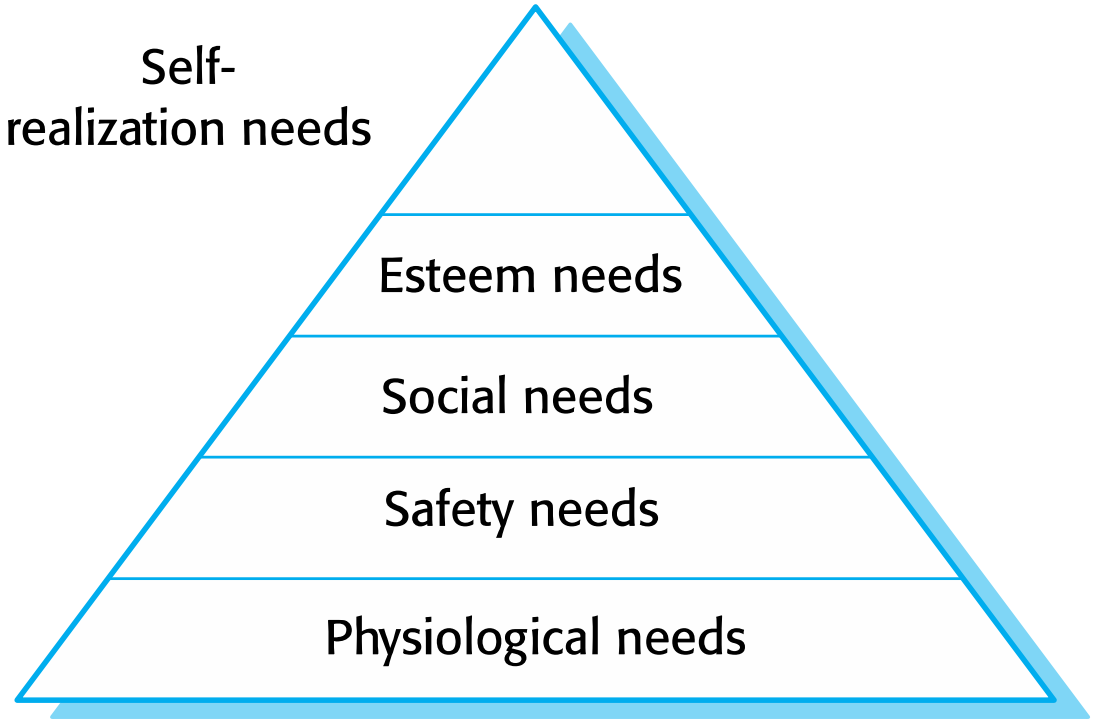
People management factors

- ✧ Consistency
  - Team members should all be treated in a comparable way without favourites or discrimination.
- ✧ Respect
  - Different team members have different skills and these differences should be respected.
- ✧ Inclusion
  - Involve all team members and make sure that people's views are considered.
- ✧ Honesty
  - You should always be honest about what is going well and what is going badly in a project.

Motivating people

- An important role of a manager is to motivate the people working on a project.
- Motivation means organizing the work and the working environment to encourage people to work effectively.
  - If people are not motivated, they will not be interested in the work they are doing. They will work slowly, be more likely to make mistakes and will not contribute to the broader goals of the team or the organization.
- Motivation is a complex issue but it appears that their are different types of motivation based on:
  - Basic needs (e.g. food, sleep, etc.);
  - Personal needs (e.g. respect, self-esteem);
  - Social needs (e.g. to be accepted as part of a group).

Human needs hierarchy



Need satisfaction

- In software development groups, basic physiological and safety needs are not an issue.
- Social
  - Provide communal facilities;
  - Allow informal communications e.g. via social networking
- Esteem
  - Recognition of achievements;
  - Appropriate rewards.
- Self-realization
  - Training - people want to learn more;
  - Responsibility.

Case study: Individual motivation

Alice is a software project manager working in a company that develops alarm systems. This company wishes to enter the growing market of assistive technology to help elderly and disabled people live independently. Alice has been asked to lead a team of 6 developers than can develop new products based around the company's alarm technology.

Alice's assistive technology project starts well. Good working relationships develop within the team and creative new ideas are developed. The team decides to develop a peer-to-peer messaging system using digital televisions linked to the alarm network for communications. However, some months into the project, Alice notices that Dorothy, a hardware design expert, starts coming into work late, the quality of her work deteriorates and, increasingly, that she does not appear to be communicating with other members of the team.
Alice talks about the problem informally with other team members to try to find out if Dorothy's personal circumstances have changed, and if this might be affecting her work. They don't know of anything, so Alice decides to talk with Dorothy to try to understand the problem.

Case study: Individual motivation

After some initial denials that there is a problem, Dorothy admits that she has lost interest in the job. She expected that she would be able to develop and use her hardware interfacing skills. However, because of the product direction that has been chosen, she has little opportunity for this. Basically, she is working as a C programmer with other team members.

Although she admits that the work is challenging, she is concerned that she is not developing her interfacing skills. She is worried that finding a job that involves hardware interfacing will be difficult after this project. Because she does not want to upset the team by revealing that she is thinking about the next project, she has decided that it is best to minimize conversation with them.

Comments on case study

- If you don’t sort out the problem of unacceptable work, the other group members will become dissatisfied and feel that they are doing an unfair share of the work.
- Personal difficulties affect motivation because people can’t concentrate on their work. They need time and support to resolve these issues, although you have to make clear that they still have a responsibility to their employer.
- Alice gives Dorothy more design autonomy and organizes training courses in software engineering that will give her more opportunities after her current project has finished.

Personality types

- The needs hierarchy is almost certainly an over-simplification of motivation in practice.
- Motivation should also take into account different personality types:
  - Task-oriented people, who are motivated by the work they do. In software engineering.
  - Interaction-oriented people, who are motivated by the presence and actions of co-workers.
  - Self-oriented people, who are principally motivated by personal success and recognition.

Personality types

- ✧ Task-oriented.
  - The motivation for doing the work is the work itself;
- ✧ Self-oriented.
  - The work is a means to an end which is the achievement of individual goals - e.g. to get rich, to play tennis, to travel etc.;
- ✧ Interaction-oriented
  - The principal motivation is the presence and actions of co-workers. People go to work because they like to go to work.

Motivation balance

- Individual motivations are made up of elements of each class.
- The balance can change depending on personal circumstances and external events.
- However, people are not just motivated by personal factors but also by being part of a group and culture.
- People go to work because they are motivated by the people that they work with.

Teamwork

Teamwork

- Most software engineering is a group activity
  - The development schedule for most non-trivial software projects is such that they cannot be completed by one person working alone.
- A good group is cohesive and has a team spirit. The people involved are motivated by the success of the group as well as by their own personal goals.
- Group interaction is a key determinant of group performance.
- Flexibility in group composition is limited
  - Managers must do the best they can with available people.

Group cohesiveness

- In a cohesive group, members consider the group to be more important than any individual in it.
- The advantages of a cohesive group are:
  - Group quality standards can be developed by the group members.
  - Team members learn from each other and get to know each other's work; Inhibitions caused by ignorance are reduced.
  - Knowledge is shared. Continuity can be maintained if a group member leaves.
  - Refactoring and continual improvement is encouraged. Group members work collectively to deliver high quality results and fix problems, irrespective of the individuals who originally created the design or program.

Team spirit

Alice, an experienced project manager, understands the importance of creating a cohesive group. As they are developing a new product, she takes the opportunity of involving all group members in the product specification and design by getting them to discuss possible technology with elderly members of their families. She also encourages them to bring these family members to meet other members of the development group.

Alice also arranges monthly lunches for everyone in the group. These lunches are an opportunity for all team members to meet informally, talk around issues of concern, and get to know each other. At the lunch, Alice tells the group what she knows about organizational news, policies, strategies, and so forth. Each team member then briefly summarizes what they have been doing and the group discusses a general topic, such as new product ideas from elderly relatives.

Every few months, Alice organizes an ‘away day’ for the group where the team spends two days on ‘technology updating’. Each team member prepares an update on a relevant technology and presents it to the group. This is an off-site meeting in a good hotel and plenty of time is scheduled for discussion and social interaction.

The effectiveness of a team

- The people in the group
  - You need a mix of people in a project group as software development involves diverse activities such as negotiating with clients, programming, testing and documentation.
- The group organization
  - A group should be organized so that individuals can contribute to the best of their abilities and tasks can be completed as expected.
- Technical and managerial communications
  - Good communications between group members, and between the software engineering team and other project stakeholders, is essential.

Selecting group members

- A manager or team leader's job is to create a cohesive group and organize their group so that they can work together effectively.
- This involves creating a group with the right balance of technical skills and personalities, and organizing that group so that the members work together effectively.

Assembling a team

- ✧ May not be possible to appoint the ideal people to work on a project
  - Project budget may not allow for the use of highly-paid staff;
  - Staff with the appropriate experience may not be available;
  - An organisation may wish to develop employee skills on a software project.
- ✧ Managers have to work within these constraints especially when there are shortages of trained staff.

Group composition

- Group composed of members who share the same motivation can be problematic
  - Task-oriented - everyone wants to do their own thing;
  - Self-oriented - everyone wants to be the boss;
  - Interaction-oriented - too much chatting, not enough work.
- An effective group has a balance of all types.
- This can be difficult to achieve software engineers are often task-oriented.
- Interaction-oriented people are very important as they can detect and defuse tensions that arise.

Group composition

In creating a group for assistive technology development, Alice is aware of the importance of selecting members with complementary personalities. When interviewing potential group members, she tried to assess whether they were task-oriented, self-oriented, or interaction-oriented. She felt that she was primarily a self-oriented type because she considered the project to be a way of getting noticed by senior management and possibly promoted. She therefore looked for one or perhaps two interaction-oriented personalities, with task-oriented individuals to complete the team. The final assessment that she arrived at was:

Alice—self-oriented
Brian—task-oriented
Bob—task-oriented
Carol—interaction-oriented
Dorothy—self-oriented
Ed—interaction-oriented
Fred—task-oriented

Group organization

- Small software engineering groups are usually organised informally without a rigid structure.
- For large projects, there may be a hierarchical structure where different groups are responsible for different sub-projects.
- Agile development is always based around an informal group on the principle that formal structure inhibits information exchange

Group communications

- Good communications are essential for effective group working.
- Information must be exchanged on the status of work, design decisions and changes to previous decisions.
- Good communications also strengthens group cohesion as it promotes understanding.

Group communications

- Group size
  - The larger the group, the harder it is for people to communicate with other group members.
- Group structure
  - Communication is better in informally structured groups than in hierarchically structured groups.
- Group composition
  - Communication is better when there are different personality types in a group and when groups are mixed rather than single sex.
- The physical work environment
  - Good workplace organisation can help encourage communications.

Chapter 23 – Project planning

Topics covered

- ✧ Software pricing
- ✧ Plan-driven development
- ✧ Project scheduling
- ✧ Agile planning
- ✧ Estimation techniques
- ✧ COCOMO cost modeling

Project planning

- Project planning involves breaking down the work into parts and assign these to project team members, anticipate problems that might arise and prepare tentative solutions to those problems.
- The project plan, which is created at the start of a project, is used to communicate how the work will be done to the project team and customers, and to help assess progress on the project.

Planning stages

- At the proposal stage, when you are bidding for a contract to develop or provide a software system.
- During the project startup phase, when you have to plan who will work on the project, how the project will be broken down into increments, how resources will be allocated across your company, etc.
- Periodically throughout the project, when you modify your plan in the light of experience gained and information from monitoring the progress of the work.

Proposal planning

- Planning may be necessary with only outline software requirements.
- The aim of planning at this stage is to provide information that will be used in setting a price for the system to customers.
- Project pricing involves estimating how much the software will cost to develop, taking factors such as staff costs, hardware costs, software costs, etc. into account

Project startup planning

- At this stage, you know more about the system requirements but do not have design or implementation information
- Create a plan with enough detail to make decisions about the project budget and staffing.
  - This plan is the basis for project resource allocation
- The startup plan should also define project monitoring mechanisms
- A startup plan is still needed for agile development to allow resources to be allocated to the project

Development planning

- The project plan should be regularly amended as the project progresses and you know more about the software and its development
- The project schedule, cost-estimate and risks have to be regularly revised

Plan-driven development

Plan-driven development

- Plan-driven or plan-based development is an approach to software engineering where the development process is planned in detail.
  - Plan-driven development is based on engineering project management techniques and is the 'traditional' way of managing large software development projects.
- A project plan is created that records the work to be done, who will do it, the development schedule and the work products.
- Managers use the plan to support project decision making and as a way of measuring progress.

Plan-driven development – pros and cons

- The arguments in favor of a plan-driven approach are that early planning allows organizational issues (availability of staff, other projects, etc.) to be closely taken into account, and that potential problems and dependencies are discovered before the project starts, rather than once the project is underway.
- The principal argument against plan-driven development is that many early decisions have to be revised because of changes to the environment in which the software is to be developed and used.

Project plans

- In a plan-driven development project, a project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work.
- Plan sections
  - Introduction
  - Project organization
  - Risk analysis
  - Hardware and software resource requirements
  - Work breakdown
  - Project schedule
  - Monitoring and reporting mechanisms

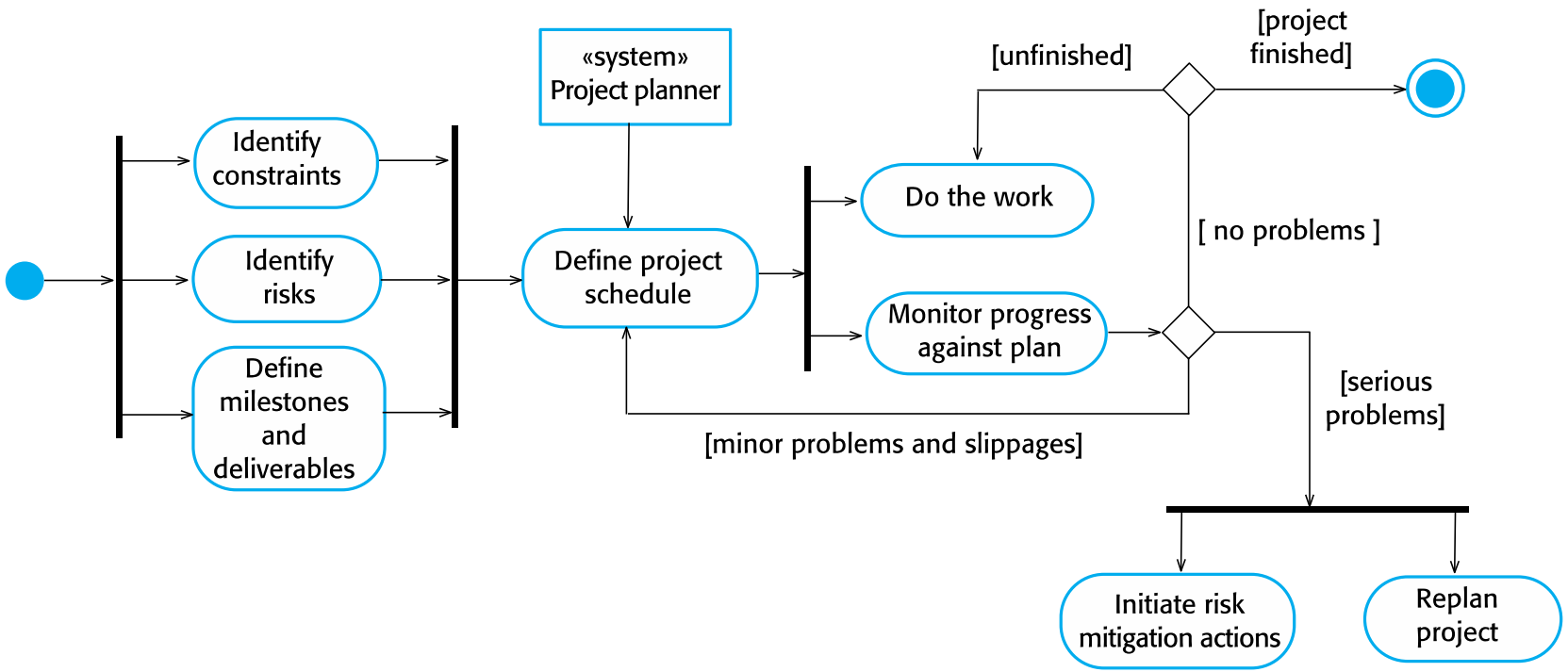
Project plan supplements

Plan	Description
Configuration management plan	Describes the configuration management procedures and structures to be used.
Deployment plan	Describes how the software and associated hardware (if required) will be deployed in the customer's environment. This should include a plan for migrating data from existing systems.
Maintenance plan	Predicts the maintenance requirements, costs, and effort.
Quality plan	Describes the quality procedures and standards that will be used in a project.
Validation plan	Describes the approach, resources, and schedule used for system validation.

The planning process

- Project planning is an iterative process that starts when you create an initial project plan during the project startup phase.
- Plan changes are inevitable.
  - As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes.
  - Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.

The project planning process



Planning assumptions

- You should make realistic rather than optimistic assumptions when you are defining a project plan.
- Problems of some description always arise during a project, and these lead to project delays.
- Your initial assumptions and scheduling should therefore take unexpected problems into account.
- You should include contingency in your plan so that if things go wrong, then your delivery schedule is not seriously disrupted.

Risk mitigation

- If there are serious problems with the development work that are likely to lead to significant delays, you need to initiate risk mitigation actions to reduce the risks of project failure.
- In conjunction with these actions, you also have to re-plan the project.
- This may involve renegotiating the project constraints and deliverables with the customer. A new schedule of when work should be completed also has to be established and agreed with the customer.

Project scheduling

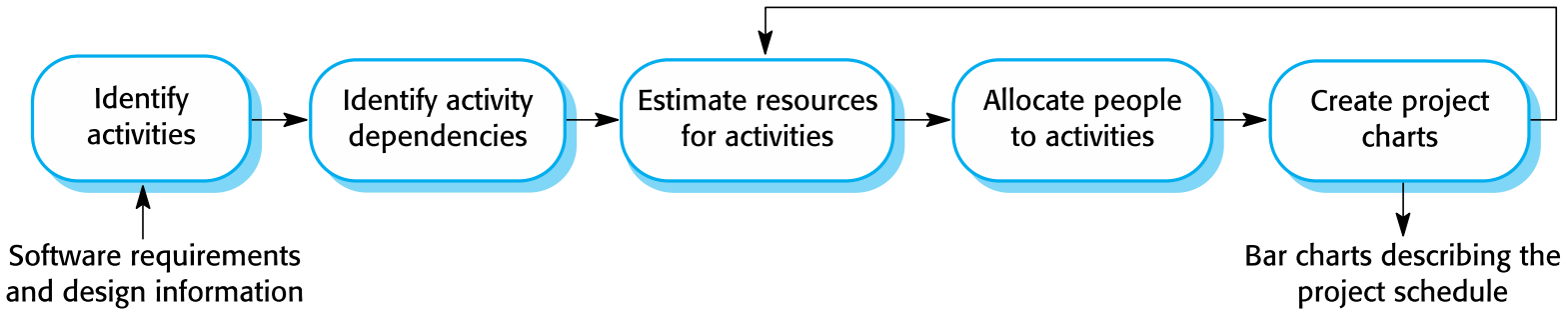
Project scheduling

- Project scheduling is the process of deciding how the work in a project will be organized as separate tasks, and when and how these tasks will be executed.
- You estimate the calendar time needed to complete each task, the effort required and who will work on the tasks that have been identified.
- You also have to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be.

Project scheduling activities

- Split project into tasks and estimate time and resources required to complete each task.
- Organize tasks concurrently to make optimal use of workforce.
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Dependent on project managers intuition and experience.

The project scheduling process



Scheduling problems

- Estimating the difficulty of problems and hence the cost of developing a solution is hard.
- Productivity is not proportional to the number of people working on a task.
- Adding people to a late project makes it later because of communication overheads.
- The unexpected always happens. Always allow contingency in planning.

Schedule presentation

- Graphical notations are normally used to illustrate the project schedule.
- These show the project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- Calendar-based
  - Bar charts are the most commonly used representation for project schedules. They show the schedule as activities or resources against time.
- Activity networks
  - Show task dependencies

Project activites

- Project activities (tasks) are the basic planning element. Each activity has:
  - a duration in calendar days or months,
  - an effort estimate, which shows the number of person-days or person-months to complete the work,
  - a deadline by which the activity should be complete,
  - a defined end-point, which might be a document, the holding of a review meeting, the successful execution of all tests, etc.

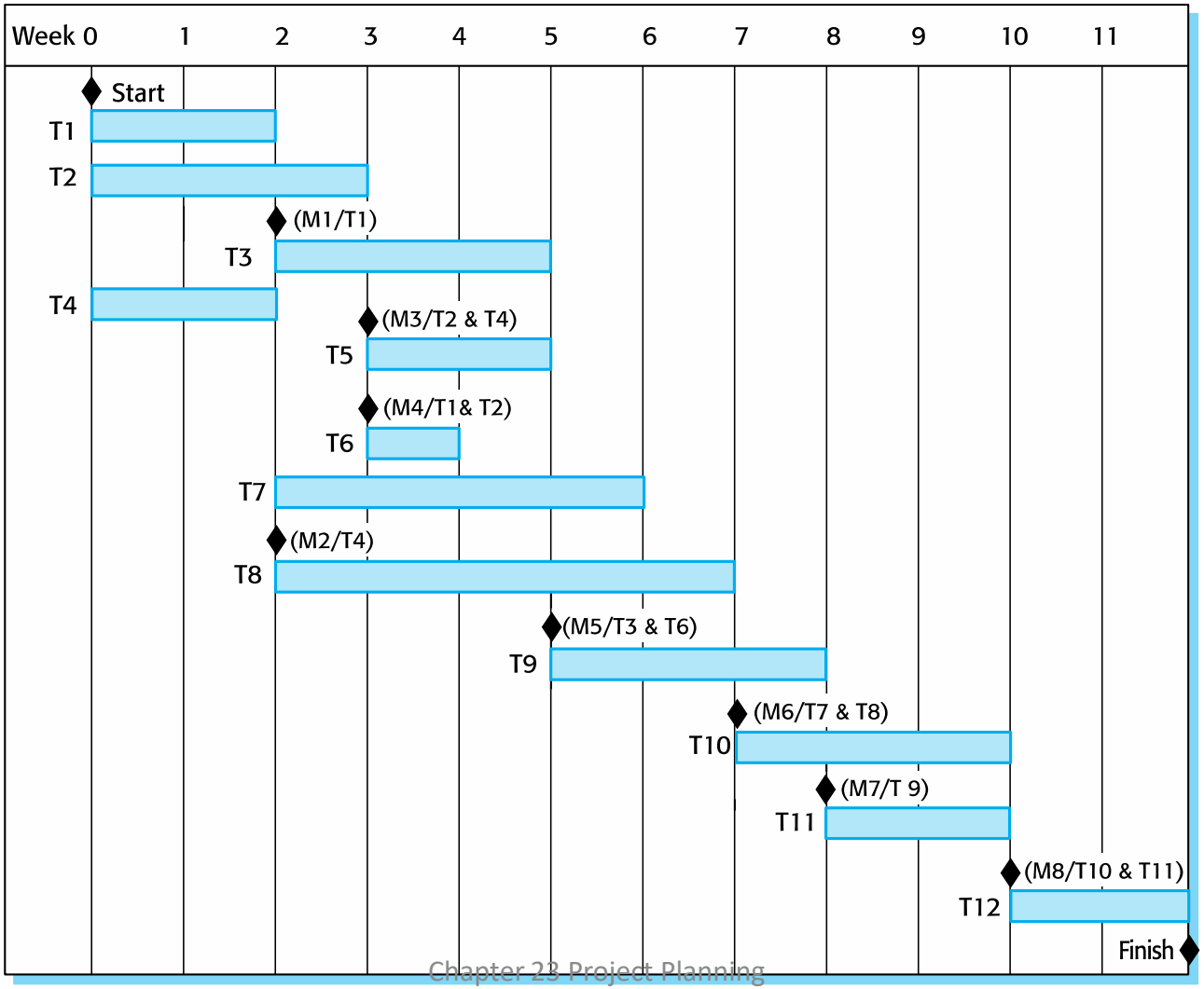
Milestones and deliverables

- Milestones are points in the schedule against which you can assess progress, for example, the handover of the system for testing.
- Deliverables are work products that are delivered to the customer, e.g. a requirements document for the system.

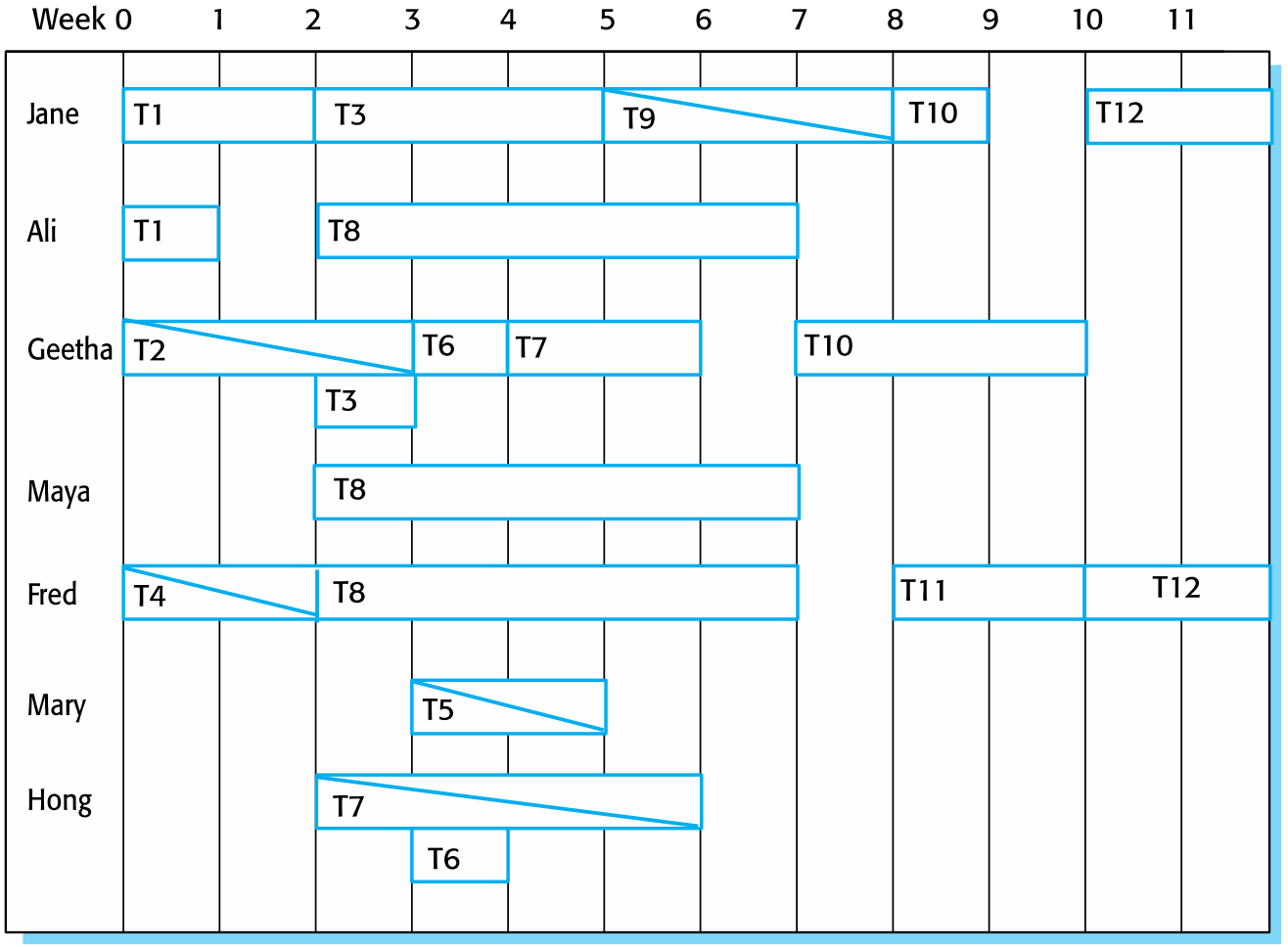
Tasks, durations, and dependencies

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Activity bar chart



Staff allocation chart



Agile planning

Agile planning

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments.
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
  - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes.

Agile planning stages

- Release planning, which looks ahead for several months and decides on the features that should be included in a release of a system.
- Iteration planning, which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2-4 weeks of work for the team.

Approaches to agile planning

- Planning in Scrum
  - Covered in Chapter 3
- Based on managing a project backlog (things to be done) with daily reviews of progress and problems
- The planning game
  - Developed originally as part of Extreme Programming (XP)
  - Dependent on user stories as a measure of progress in the project

Story-based planning

- The planning game is based on user stories that reflect the features that should be included in the system.
- The project team read and discuss the stories and rank them in order of the amount of time they think it will take to implement the story.
- Stories are assigned ‘effort points’ reflecting their size and difficulty of implementation
- The number of effort points implemented per day is measured giving an estimate of the team’s ‘velocity’
- This allows the total effort required to implement the system to be estimated

The planning game

Release and iteration planning

- Release planning involves selecting and refining the stories that will reflect the features to be implemented in a release of a system and the order in which the stories should be implemented.
- Stories to be implemented in each iteration are chosen, with the number of stories reflecting the time to deliver an iteration (usually 2 or 3 weeks).
- The team's velocity is used to guide the choice of stories so that they can be delivered within an iteration.

Task allocation

- During the task planning stage, the developers break down stories into development tasks.
  - A development task should take 4–16 hours.
  - All of the tasks that must be completed to implement all of the stories in that iteration are listed.
  - The individual developers then sign up for the specific tasks that they will implement.
- Benefits of this approach:
  - The whole team gets an overview of the tasks to be completed in an iteration.
  - Developers have a sense of ownership in these tasks and this is likely to motivate them to complete the task.

Software delivery

- A software increment is always delivered at the end of each project iteration.
- If the features to be included in the increment cannot be completed in the time allowed, the scope of the work is reduced.
- The delivery schedule is never extended.

Agile planning difficulties

- Agile planning is reliant on customer involvement and availability.
- This can be difficult to arrange, as customer representatives sometimes have to prioritize other work and are not available for the planning game.
- Furthermore, some customers may be more familiar with traditional project plans and may find it difficult to engage in an agile planning process.

Agile planning applicability

- Agile planning works well with small, stable development teams that can get together and discuss the stories to be implemented.
- However, where teams are large and/or geographically distributed, or when team membership changes frequently, it is practically impossible for everyone to be involved in the collaborative planning that is essential for agile project management.

Estimation techniques

Software: What to Estimate?

- Size of software
  - Lines of Code – KLOC
    - E.g. 3,20,000 Lines
  - Function Points – FP
- Effort
  - Person Months – PM
  - Effort = Size / Productivity
    - Effort = 3,20,000 LOC / 8,000 LOC/PM = 40 person months
- Duration
  - Distribute effort over time
    - Look at available time, available talent
    - If you have 5 developers, then, Duration = 40 pm / 5p = 8 months
    - If you have 4 months to deliver, Developers = 40 pm / 4m = 10 persons

Size Estimation Approaches

- Lines Of Code (LoC) based
- Function Points (FP) based
- Process based
- Use-case based

Three Points Based Estimation (Aka PERT)

PERT = Program Evaluation & Review Technique

- Take three estimates
  - Best Case 75m (Optimistic) O
  - Most Likely 90m (Most Likely) ML
  - Worst Case 180m (Pessimistic) P
  - Estimate = (O + 4 ML + P) / 6
    - E = (75 + 4x90 + 180) / 6 = 615/6 = 102.5 m
  - Standard Deviation = [(P - O) / 6]^2
    - = [(180-75)/6]^2 = [105/6]^2 = [17.5]^2 = 306.25

LOC based Estimation

- Decompose the software into modules/functions
- Estimate KLOC based on experience/past data/similar projects
- Add up
- Size = 33,200 LOC, Productivity = 620 LOC/pm, Labour = 8000 $/Month, Compute effort and project cost, duration with 2 developers
  - Effort = LOC/P = 33,200/620 = 53.5 pm
  - Cost = Effort x Labour = 53.5 pm x 8000 $/pm = 4,28,388 $
  - Duration = Effort / Resources = 53.5 pm / 2p = 26.75m

FP based Estimation

- FP = Function Point
- Decompose into functions and estimate size based on *inputs (EI)*, *outputs (EO)*, enquiries (Eiq), logical files (L), and interfaces (I)
- FP = count total x [0.65 + 1% of VAF]
  - FP = 50 x (0.65 * 1% of 46) = 50 x 1.11 = 55.5 FP
- Productivity = 7 FP/pm
- Effort = S/P = 55.5 / 7 = 7.92 ~ 8 pm
- Asked to finish in 4 months = developers = Effort/Time = 8/4 = 2 dev
- Asked to complete with 3 dev = Time = Effort/Team Size = 8pm/3p = 2.667m
- Cost = 8000 $/pm = 8 pm x 8000 $/pm = 64,000 $

Process based Estimation

- Decompose Requirements into functions
- For each function, estimate how much time is needed for analysis, design, coding, testing, customer interaction
- Add up, considering overheads (planning, cust comn., risk analysis)
- Effort = 46 person months
- Tricky to estimate resources

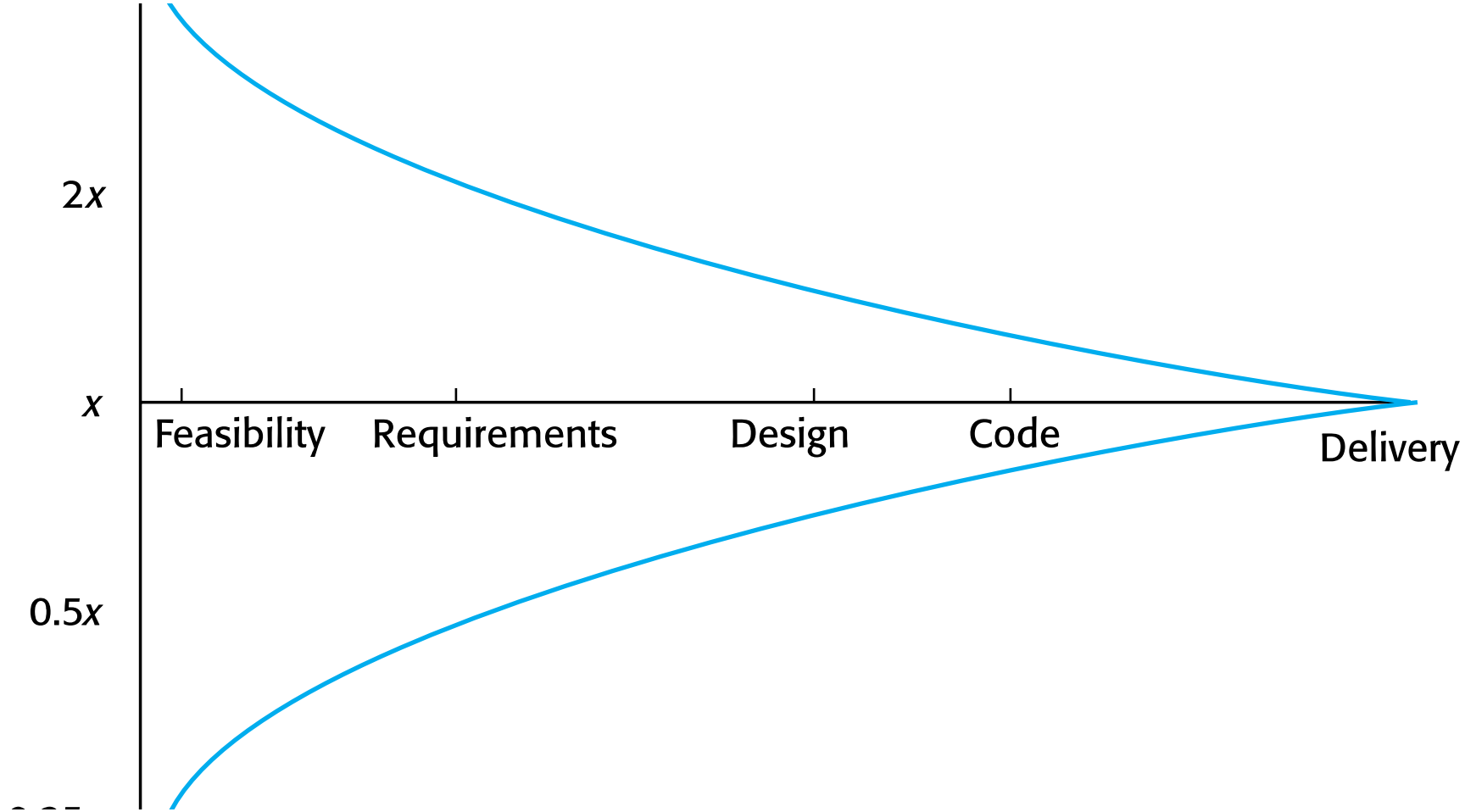
UC based Estimation

- Decompose Requirements into use cases
- LOC estimate = N x LOCavg + [(Sa/Sh – 1) x (Pa/Ph - 1)] x LOCadjust
- Size in LOC
- Effort = Size/Organization productivity
- Cost = Effort in pm x USD per pm

Estimation techniques

- Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:
  - *Experience-based techniques* The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.
  - *Algorithmic cost modeling* In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

Estimate uncertainty



COCOMO cost modeling

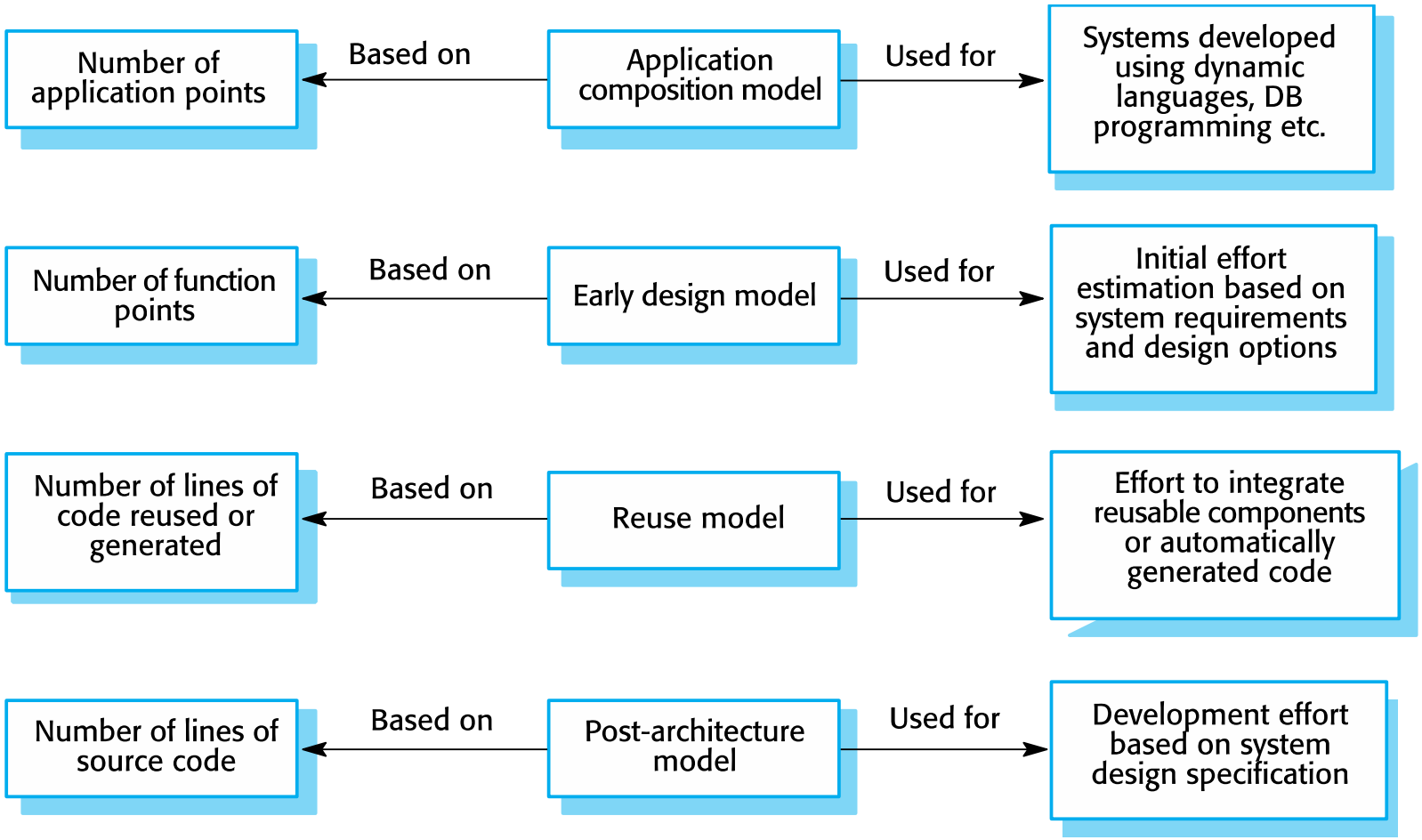
COCOMO cost modeling

- An empirical model based on project experience.
- Well-documented, 'independent' model which is not tied to a specific software vendor.
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2.
- COCOMO 2 takes into account different approaches to software development, reuse, etc.

COCOMO 2 models

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO 2 are:
  - Application composition model. Used when software is composed from existing parts.
  - Early design model. Used when requirements are available but design has not yet started.
  - Reuse model. Used to compute the effort of integrating reusable components.
  - Post-architecture model. Used once the system architecture has been designed and more information about the system is available.

COCOMO estimation models



Software Quality

What is Good Software?

- Context matters
- Quality: Considered in THREE dimension
  - Quality of the product
  - Quality of the process
  - Quality of the business environment in which product is used

Software quality attributes

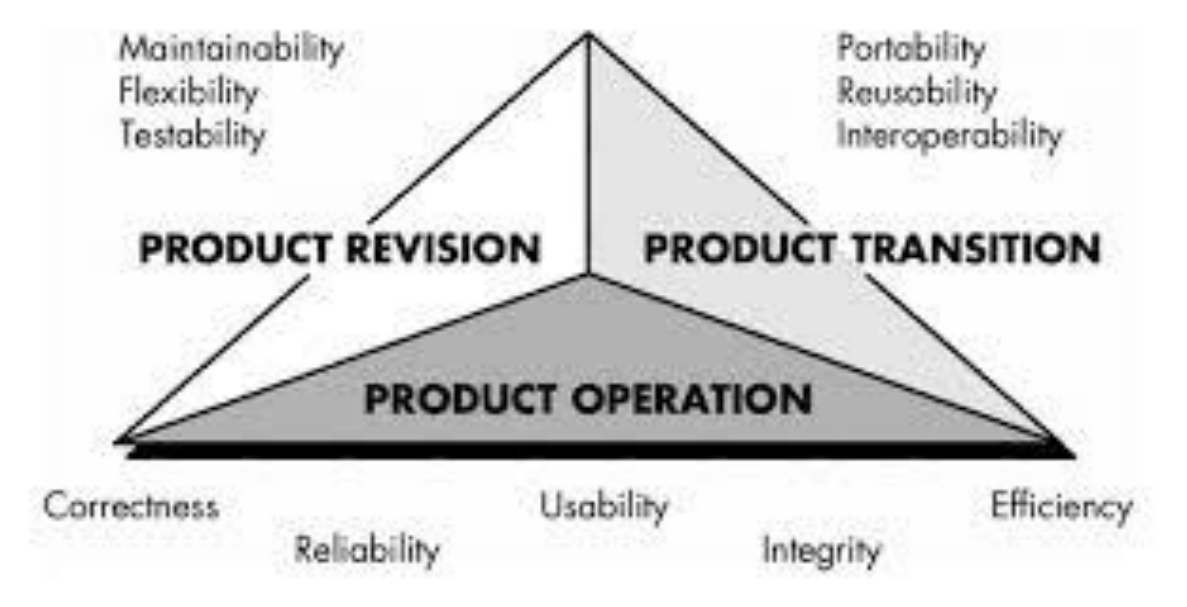
Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Quality Definitions based on Views

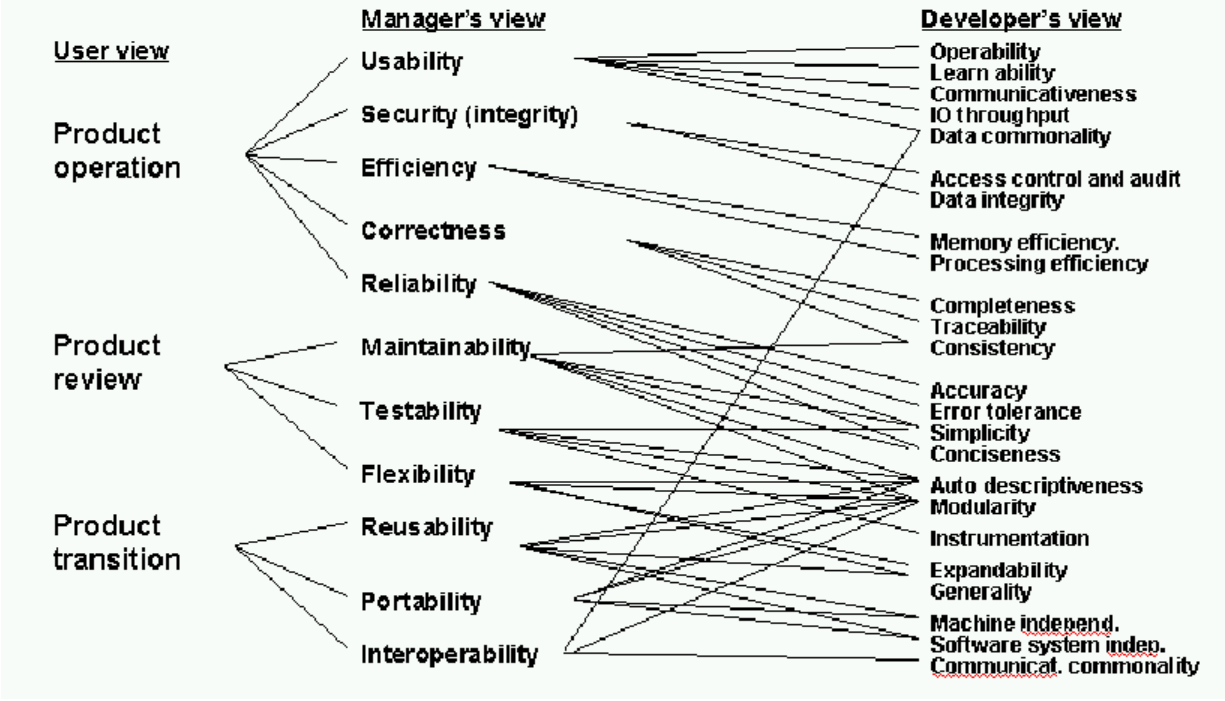
- User View: “Fitness for purpose”
- Product View: Set of good product characteristics
- Manufacturing View: Conformance to specification
- Value based View: How much the customer is willing to pay for it
- Transcendental View: Something that we can recognize but cannot define

Seven Popular Quality Models

- McCall's Model
- ISO 9126
- Boehm's Model = McCall's Model + Hardware specs
- Dromey's Model
- ISO 9001
  - Generic Quality Management System
- CMM – Capability Maturity Model
  - 5 Levels
  - CMMI – Capability Maturity Model Integrated
- SPICE
  - Software Process Improvement and Capability Determination


Maintainability
Flexibility
Testability
Portability
Reusability
Interoperability
PRODUCT REVISION
PRODUCT TRANSITION
PRODUCT OPERATION
Correctness
Reliability
Usability
Integrity
Efficiency

McCall model



ISO 9126

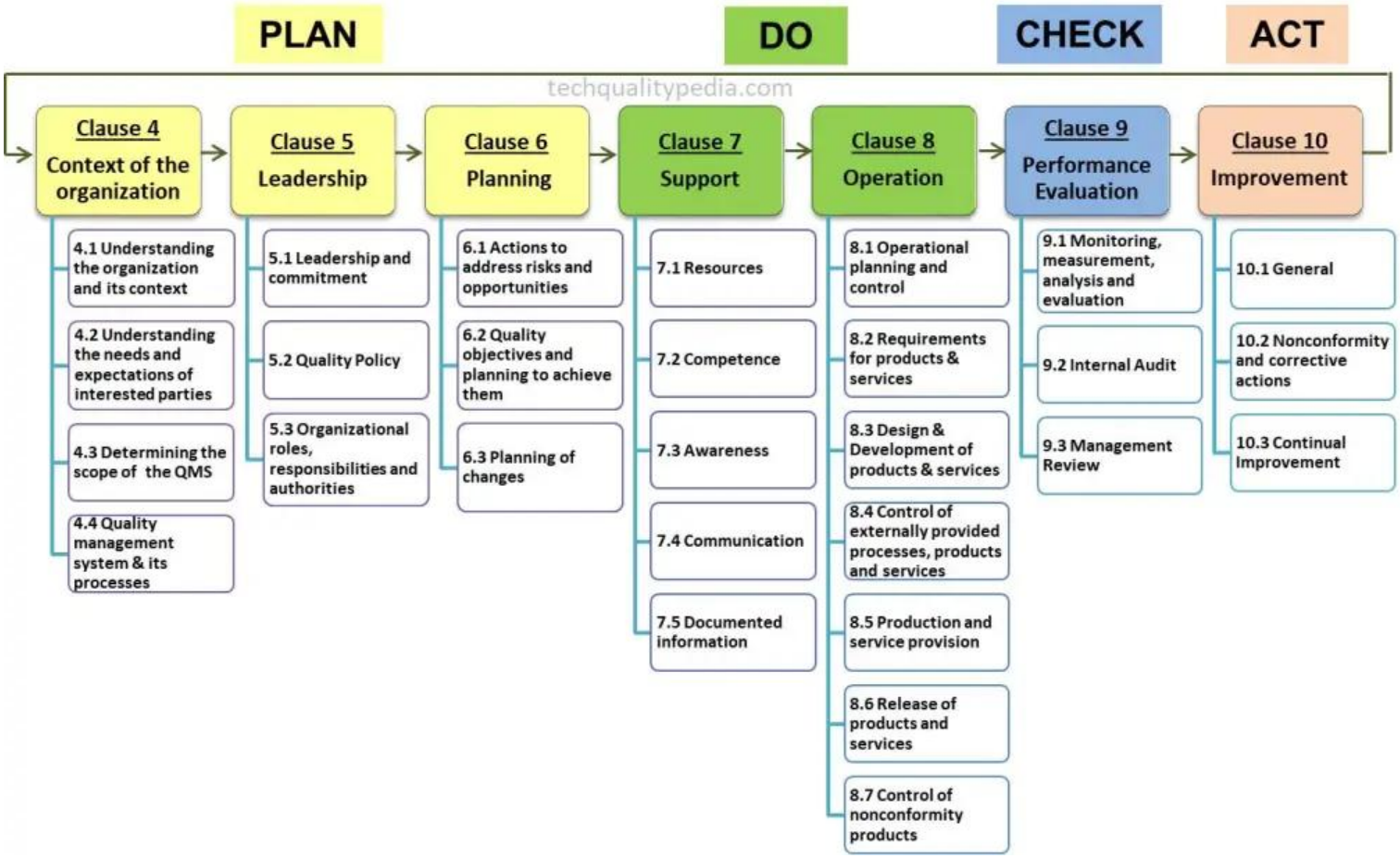
- Quality Model
- Six dimensions of quality
  - Functionality — Have features
  - Reliability — Features should be reliable
  - Usability — Reliable features should usable
  - Efficiency — Product must use minimum resources
  - Maintainable — Product should be maintainable
  - Portability — Product should work in other contexts

ISO 9001

- Standard
- Made up of 20 clauses

CLAUSES OF ISO 9001 | QUALITY MANAGEMENT SYSEM ISO 9001

(CLAUSES 4 TO 10 ARRANGED ACCORDING TO PDCA)



Software standards

Software standards

- Standards define the required attributes of a product or process. They play an important role in quality management.
- Standards may be international, national, organizational or project standards.

Importance of standards

- Encapsulation of best practice- avoids repetition of past mistakes.
- They are a framework for defining what quality means in a particular setting i.e. that organization's view of quality.
- They provide continuity - new staff can understand the organisation by understanding the standards that are used.

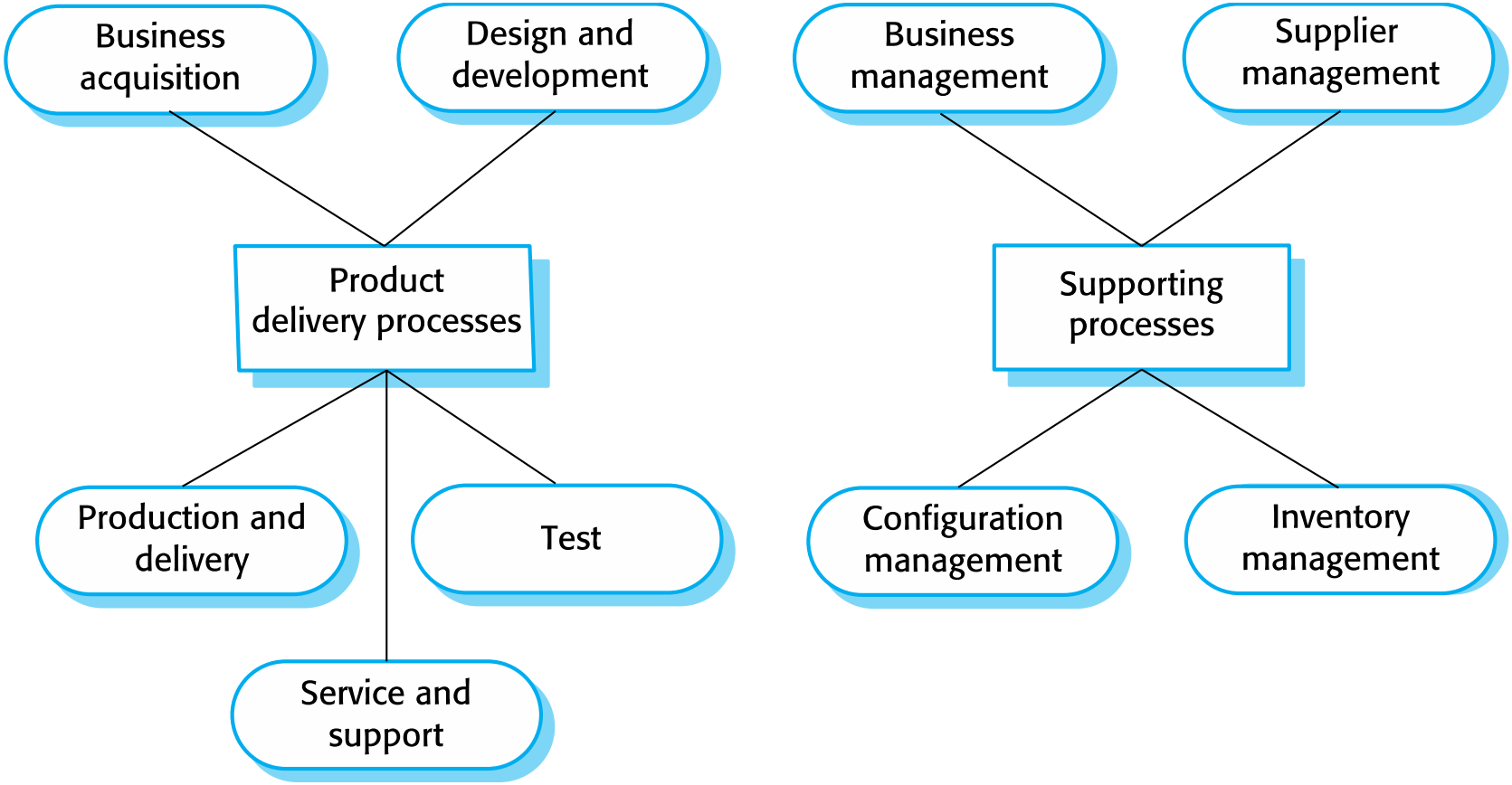
Problems with standards

- They may not be seen as relevant and up-to-date by software engineers.
- They often involve too much bureaucratic form filling.
- If they are unsupported by software tools, tedious form filling work is often involved to maintain the documentation associated with the standards.

ISO 9001 standards framework

- An international set of standards that can be used as a basis for developing quality management systems.
- ISO 9001, the most general of these standards, applies to organizations that design, develop and maintain products, including software.
- The ISO 9001 standard is a framework for developing software standards.
  - It sets out general quality principles, describes quality processes in general and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.

ISO 9001 core processes



Reviews and inspections

Reviews and inspections

- A group examines part or all of a process or system and its documentation to find potential problems.
- Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.
- There are different types of review with different objectives
  - Inspections for defect removal (product);
  - Reviews for progress assessment (product and process);
  - Quality reviews (product and standards).

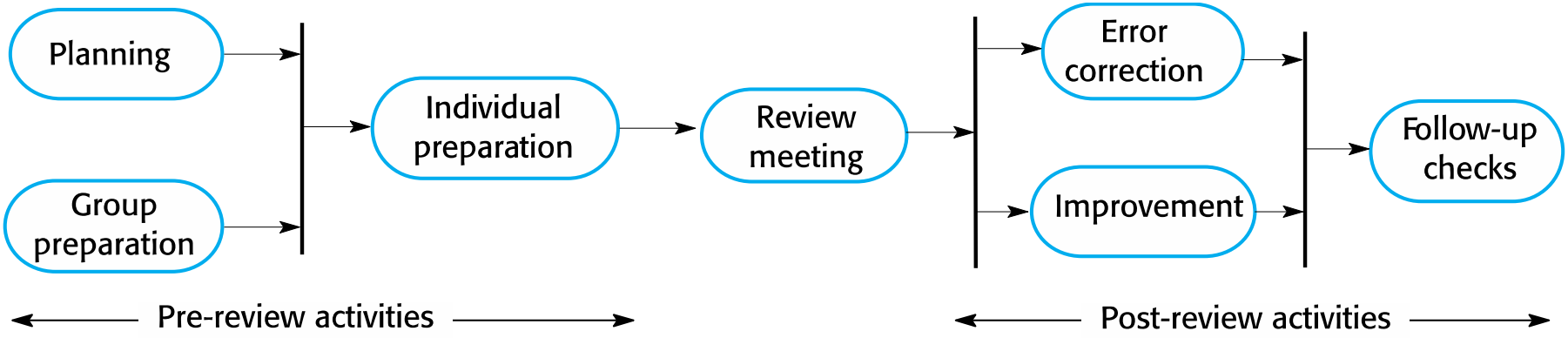
Quality reviews

- A group of people carefully examine part or all of a software system and its associated documentation.
- Code, designs, specifications, test plans, standards, etc. can all be reviewed.
- Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.

Phases in the review process

- ✧ Pre-review activities
  - Pre-review activities are concerned with review planning and review preparation
- ✧ The review meeting
  - During the review meeting, an author of the document or program being reviewed should ‘walk through’ the document with the review team.
- ✧ Post-review activities
  - These address the problems and issues that have been raised during the review meeting.

The software review process



Program inspections

- These are peer reviews where engineers examine the source of a system with the aim of discovering anomalies and defects.
- Inspections do not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design,configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Inspection checklists

- Checklist of common errors should be used to drive the inspection.
- Error checklists are programming language dependent and reflect the characteristic errors that are likely to arise in the language.
- In general, the 'weaker' the type checking, the larger the checklist.
- Examples: Initialisation, Constant naming, loop termination, array bounds, etc.

An inspection checklist (a)

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?

An inspection checklist (b)

Fault class	Inspection check
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?

Handson
Software Engineering

The Royal Service Station

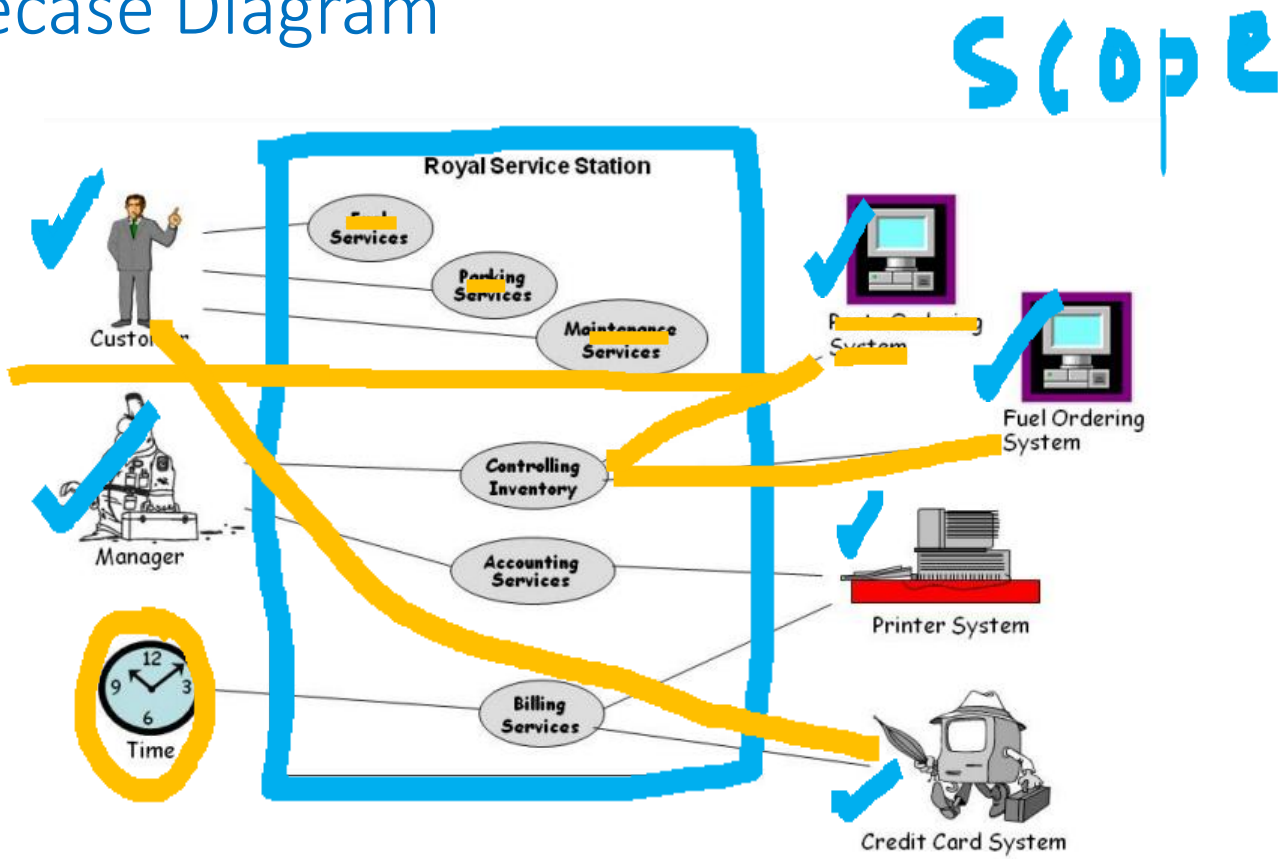
1. The Royal Service Station provides three types of services to its [redacted]: refueling, vehicle maintenance, and p[redacted]g. That is, a customer can add f[redacted] to the tank of his or her vehicle (car, motorcycle, or truck), can have the [redacted] repaired, or can park the vehicle in the station parking lot. A customer has the option to be billed automatically at the time of [redacted] (of fuel, maintenance, or parking) or to be sent a monthly [redacted]l. In either case, customers can pay using cash, [redacted], or [redacted]. Royal Service Station fuel is sold according to price per gallon, depending on whether the fuel is diesel, regular, or premium. [redacted] is priced according to the cost of parts and labor. [redacted] is sold according to daily, weekly, and monthly rates. The prices for fuel, r[redacted] services, parts, and parking may vary; only Manny, the s[redacted]r, can enter or change p[redacted]s. At his discretion, Manny may designate a [redacted] on purchases for a particular customer; this discount may vary from one customer to another. A[redacted]purchases.
2. The system must track bills on a month-to-month basis and the products and services provided by the gas station on a day-to-day basis. The results of this tracking can be reported to the station manager upon request.
3. The station manager uses the system to control inventory. The system will warn of low inventory and automatically order new parts and fuel.
4. The system will track credit history and send warning letters to customers whose payments are overdue. Bills are sent to customers on the first day of the month after the purchases are made. Payment is due on the first day of the succeeding month. Any bill not paid within 90 days of the billing date will result in cancellation of the customer's credit.
5. The system applies only to r[redacted]rs. A regular repeat customer means a customer identified by n[redacted]birthdate who uses the station's services at least once per month for at least six months.
6. The system must handle the data requirements for interfacing with other systems. A credit card system is used to process credit card transactions for products and services. The credit card system uses the card number, name, expiration date, and amount of the purchase. After receiving this information, the credit card system confirms that the transaction is approved or denied. The parts ordering system receives the part code and number of parts needed. It returns the date of parts delivery. The fuel ordering system requires a fuel order description consisting of fuel type, number of gallons, station name, and station identification code. It returns the date when the fuel will be delivered.

7. The system must record tax and related information, including tax paid by each cus[redacted]er, as well as tax per item.
8. The station manager must be able to review tax records upon demand.
9. The system will send p[redacted]ers, reminding them when their v[redacted]icl are due for maintenance. Normally, maintenance is needed every six months.
10. Customers can rent parking spaces in the station parking lot on a day-to-day basis. Ea customer must request from the system an available parking space. The station manag can view a monthly report summarizing how many parking spaces were available occupied.
11. The system maintains a repository of account information, accessible by account numb and by customer name.
12. The station manager must be able to review accounting information upon demand.
13. The system can report an analysis of prices and discounts to the station manager u request.
14. The system will automatically notify the owners of dormant accounts. That is, custo who did not make service station purchases over a two-month period will be contacte
15. The system cannot be unavailable for more than 24 hours.
16. The system must protect customer information from unauthorized access.

Requirements Summary

1. Royal Service station provides three types of services
2. The system must track bills, the product and services
3. System to control inventory
4. The system to track credit history, and payments overdue
5. The system applies only to regular repeat customer
6. The system must handle the data requirements for interfacing with other systems
7. The system must record tax and related information
8. The station must be able to review tax record upon demand
9. The system will send periodic message to customers
10. Customer can rent parking space in the station parking lot
11. The system maintain a repository of account information
12. The station manager must be able to review accounting information upon demand
13. The system can report an analysis of prices and discounts
14. The system will automatically notify the owners of dormant accounts
15. The system can not be unavailable for more than 24 hours
16. The system must protect customer information from unauthorized access

Usecase Diagram



Verify whether chosen entity is a class or not?

- What needs to be "processed" in some way?
- What items have multiple attributes?
- When do you have more than one object in a class?
- What is based on the requirements themselves, not derived from your understanding of the requirements?
- What attributes and operations are always applicable to a class or object?

Initial Grouping of Attributes and Classes: Step 1

Attributes	Classes
Personal check	Customer
Tax	Maintenance
Price	Services
Cash	Fuel
Credit card	Bill
Discounts	Purchase
	Station manager

Initial Grouping of Attributes and Classes: Step 2

Attributes	Classes
Personal check	Customer
Tax	Maintenance
Price	Services
Cash	Parking
Credit card	Fuel
Discounts	Bill
Name	Purchase
Address	Maintenance reminder
Birthdate	Station manager

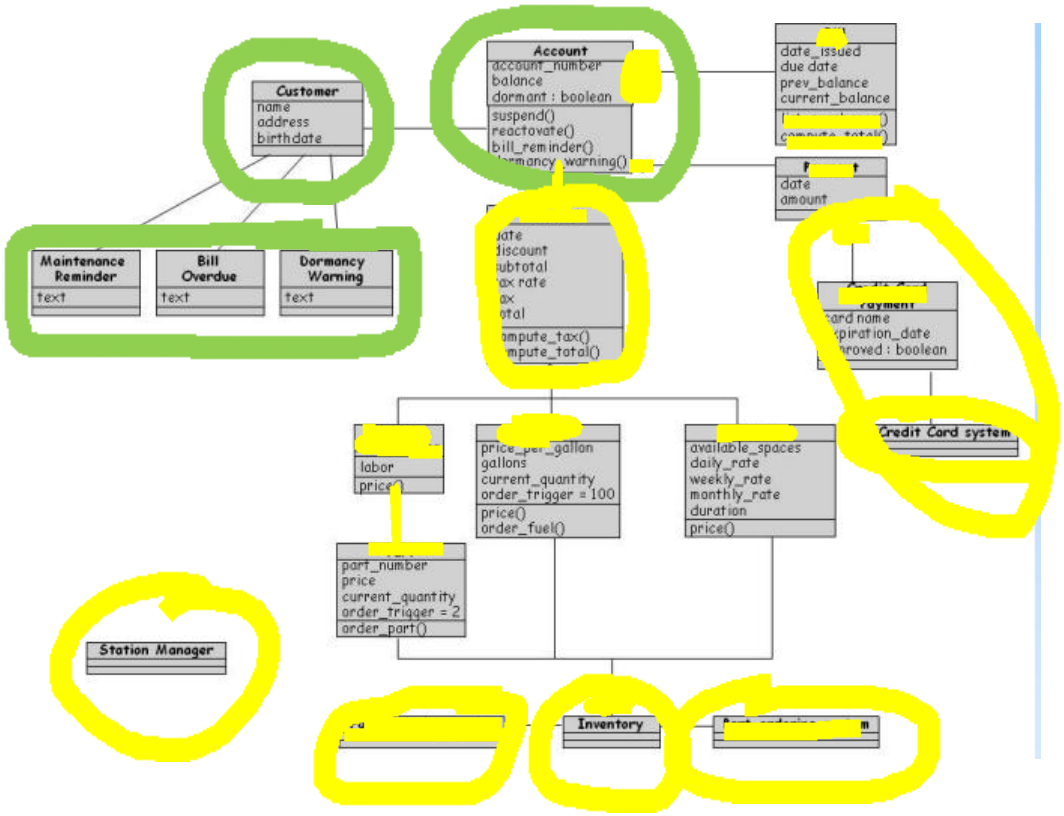
Guidelines for Identifying Behaviors

- Imperative verbs
- Passive verbs
- Actions
- Membership in
- Management or ownership
- Responsible for
- Services provided by an organization

Initial Grouping of Attributes and Classes Step 3

Attributes	Classes
Personal check	Customer
Tax	Maintenance
Price	Services
Cash	Parking
Credit card	Fuel
Discounts	Bill
Name	Purchase
Address	Maintenance reminder
Birthdate	Station manager
	[illegible]er
	[illegible]
	Parts
	Accounts
	[illegible]ry
	Credit card system
	[illegible]em
	[illegible] system

First Cut at Royal Service Station Design



Other UML Diagrams

- Class description template
- Package diagrams
- Interaction diagrams
- Sequence diagrams
- Communication diagrams
- State diagrams
- Activity diagrams

Other UML Diagrams – Class Description Template

Class name: Refuel
- **Category:** service
- **External documents:**
- **Export control:** Public
- **Cardinality:** n
- **Hierarchy:**
  - **Superclasses:** Service
- **Associations:**
  - <no rolename>: fuel in association updates

Operation name: price
- **Public member of:** Refuel
- **Documentation:**
  - // Calculates fuel final price
- **Preconditions:**
  - gallons > 0
  - Object diagram: (unspecified)
- **Semanti**
  - price = gallons * fuel_price_per_gallon
  - tax = price * purchase.tax_rate
  - Object diagram: (unspecified)
- **C rrency:** sequential

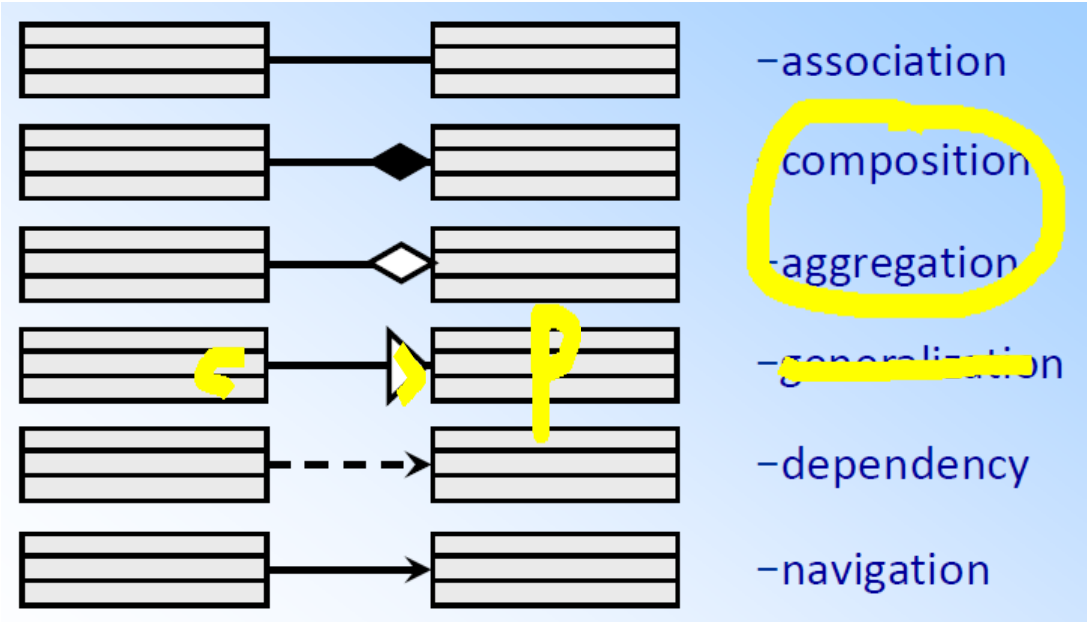
Public interfac
- **Operations:**
  - price

Private interface:
- **Attributes:**
  - gallons

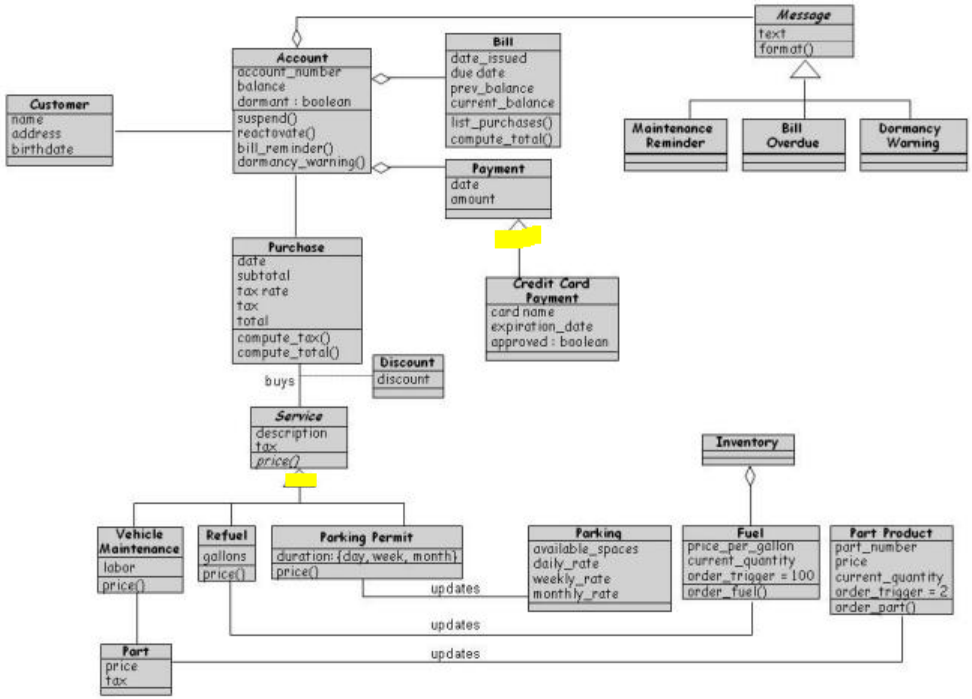
Implementation:
- **Attributes:**
  - gallons

State machine: no
Concurrency: sequential
Persistence: transient

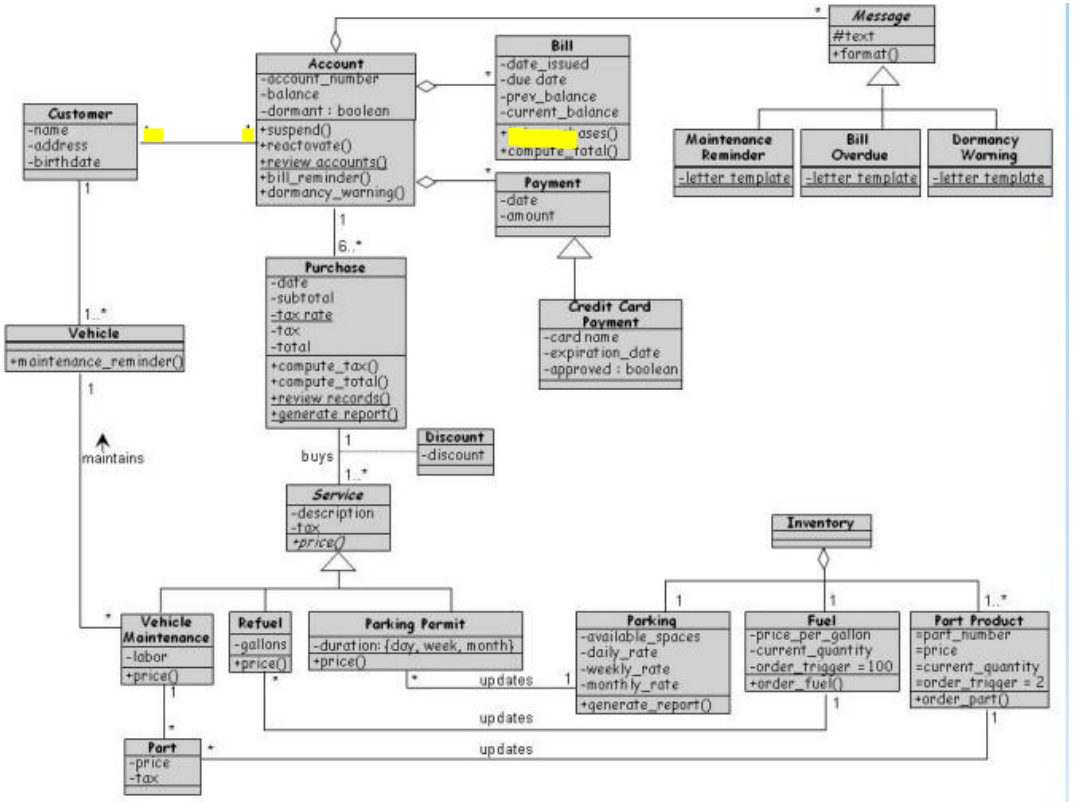
Final Cut at Royal Service Station Design



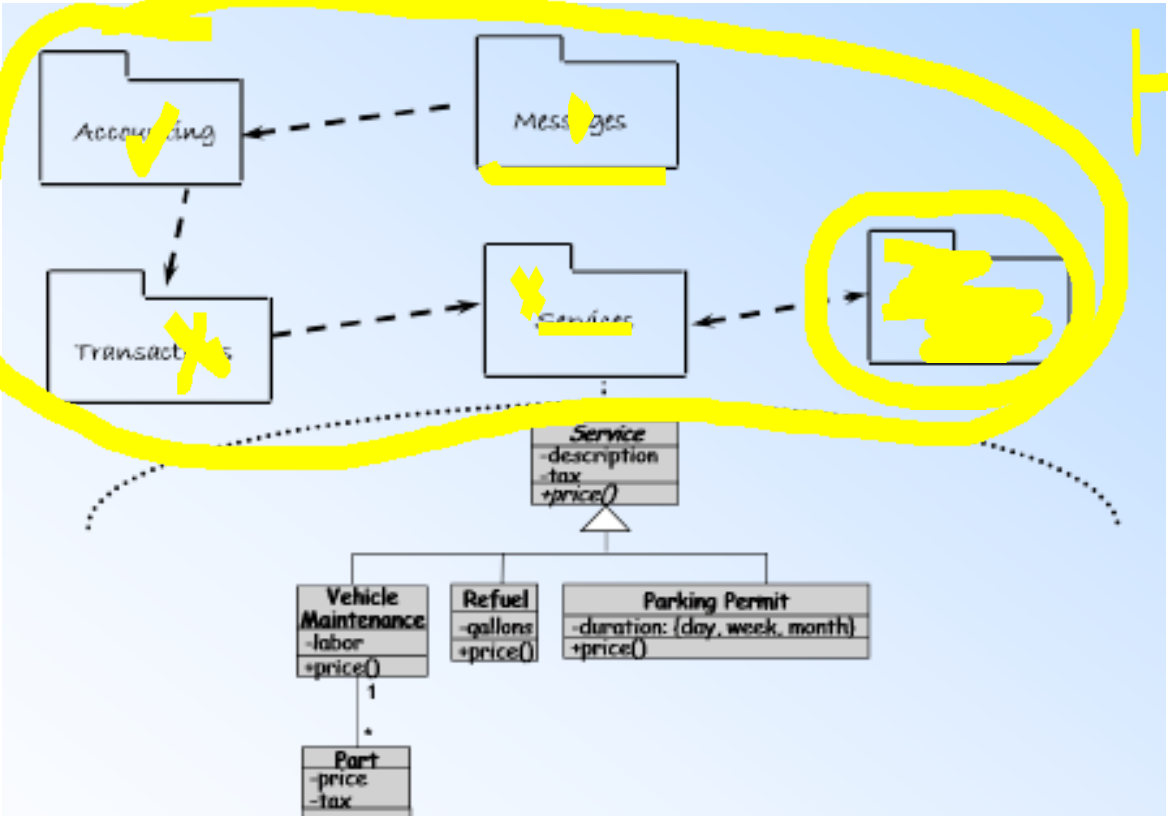
Second Cut at Royal Service Station Design



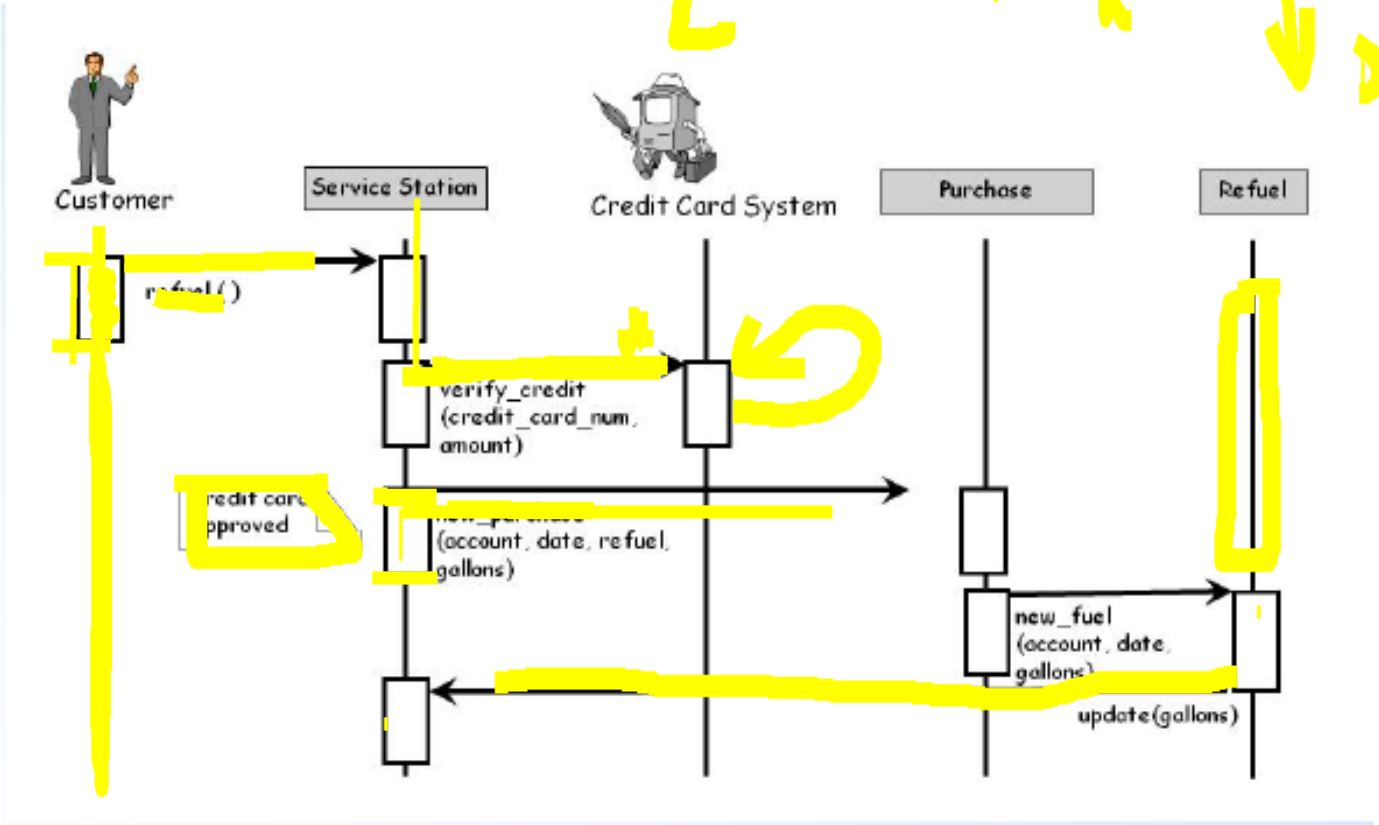
Final Cut at Royal Service Station Design



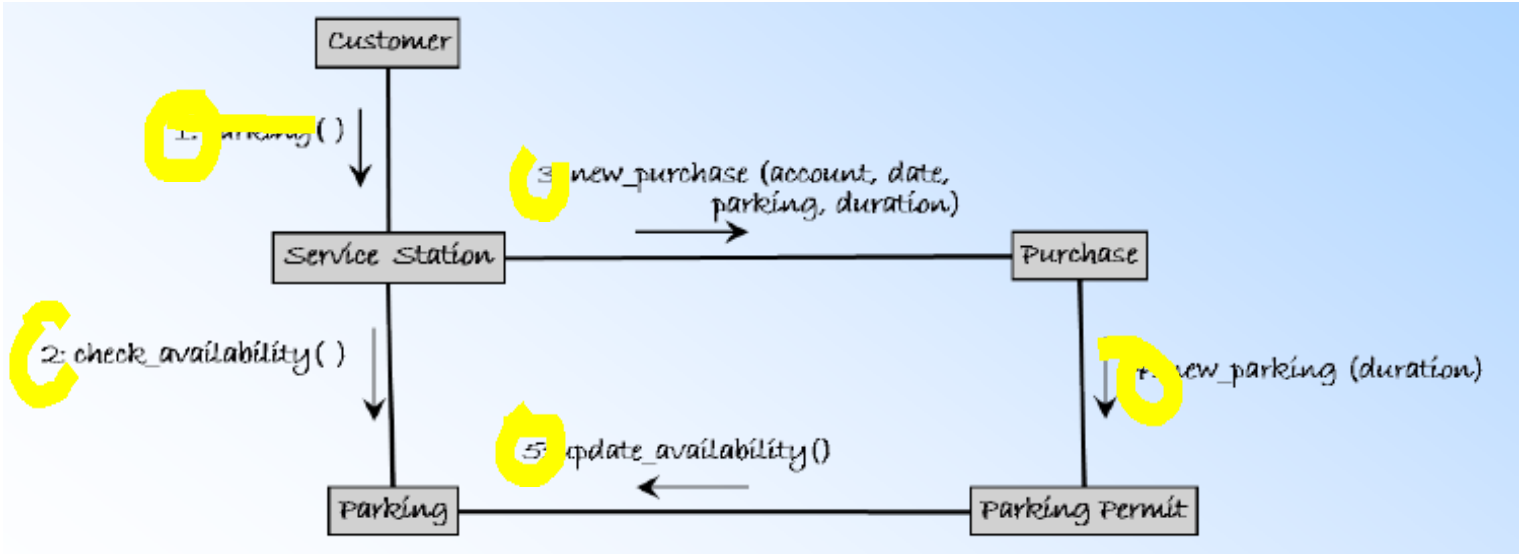
Package Diagram



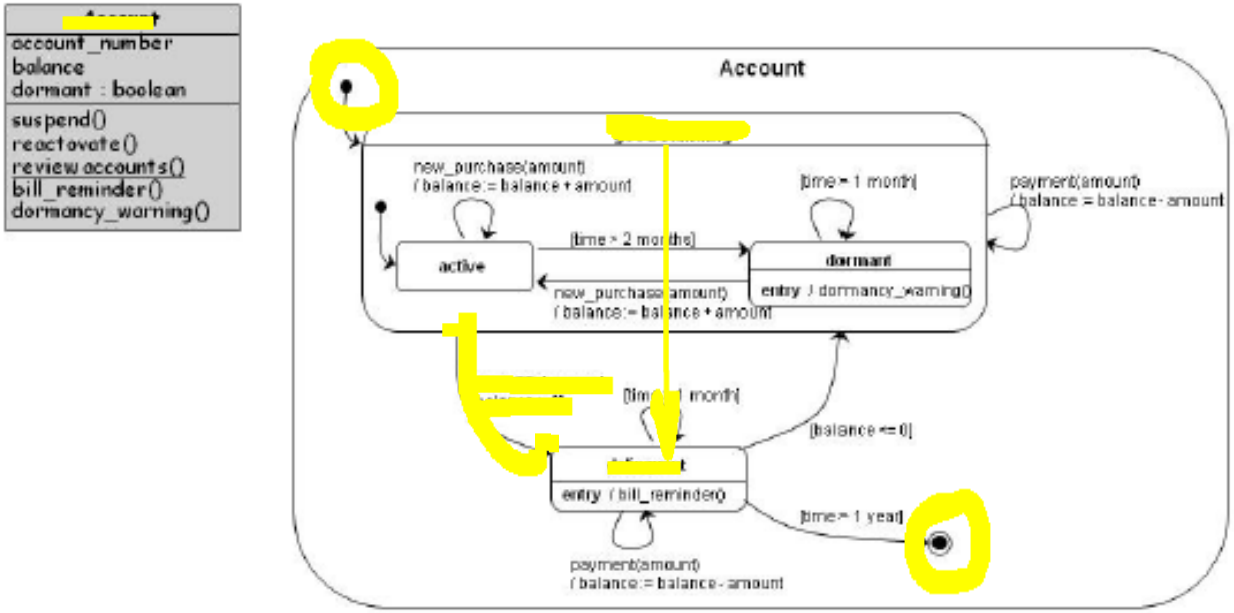
Sequence Diagram



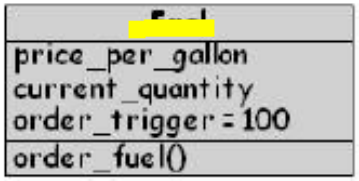
Communication Diagram

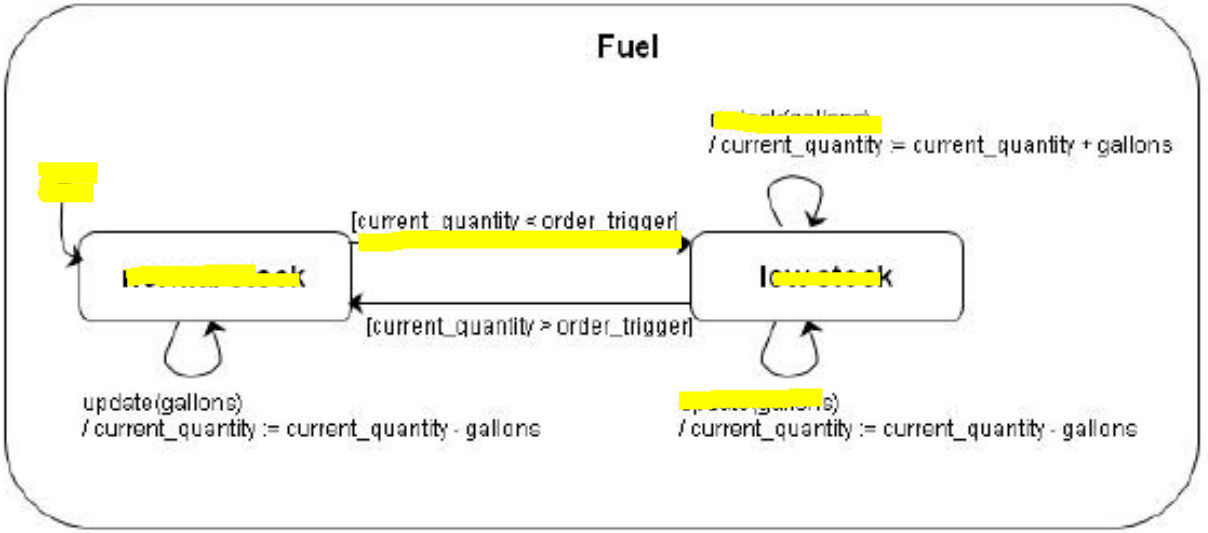


State Diagram



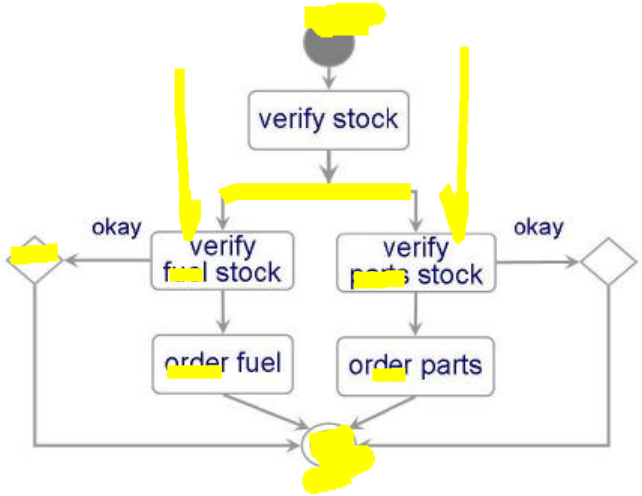
State Diagram (continued)





Activity Diagram

- An activity diagram for the *inventory* class
- It may have two decisions
  - to verify that there are enough fuel
    to verify that a part is in stock

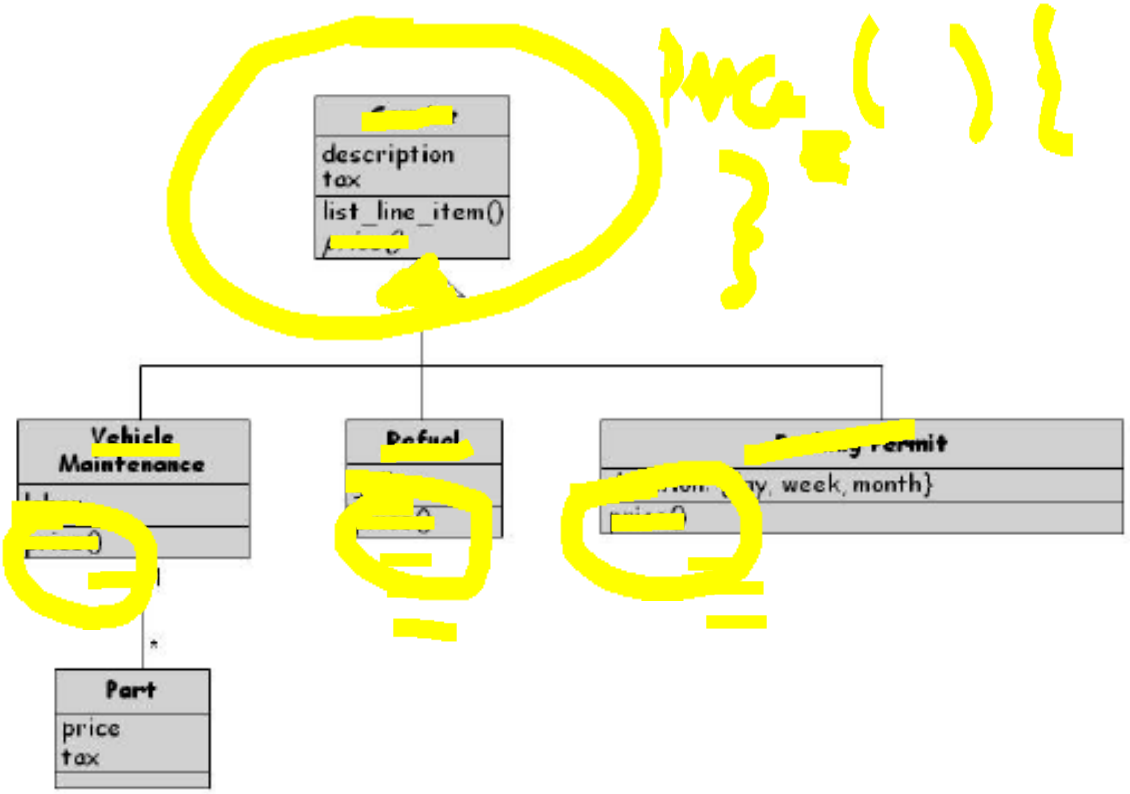


Design Patterns

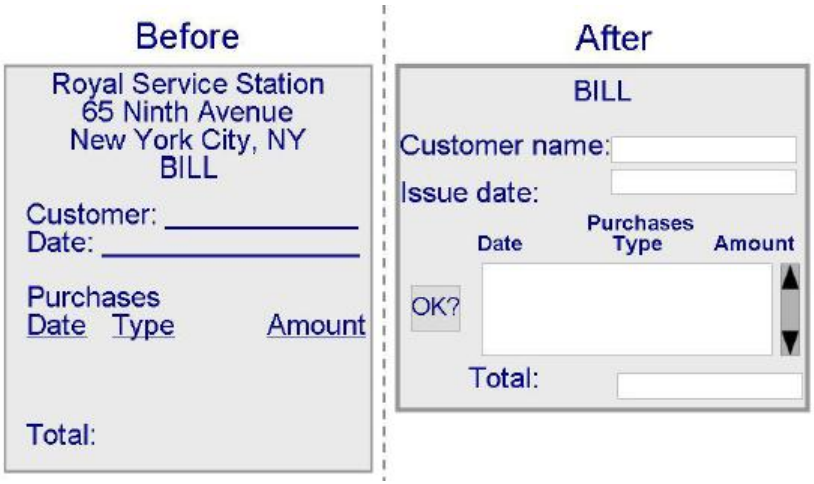
- Creational Patterns
  - Abstract Factory
  - Builder
  - Factory Method
  - Prototype
  - Singleton
- Structural Patterns
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Façade
  - Flyweight
  - Proxy
- Behavioral Patterns
  - Chain of Responsibility
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor

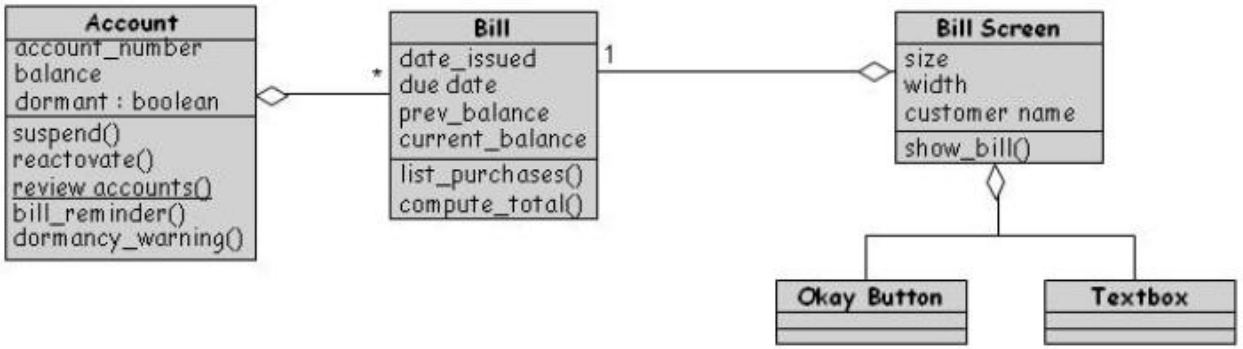
OO Design Patterns: [redacted] Pattern

Create “list_line_item()” method in Services class that prints the line item fields. This method calls a local abstract method class “[redacted]()” to print the item’s price. Each of the service subclasses [redacted]s the “[redacted]()” method to reflect how the price for that service is computed

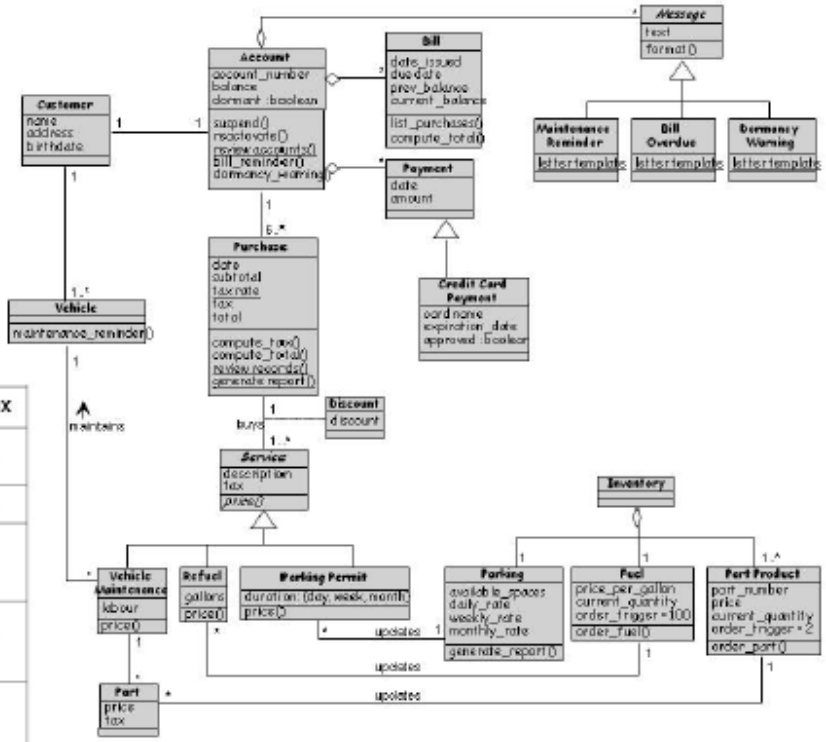


Design User Interfaces



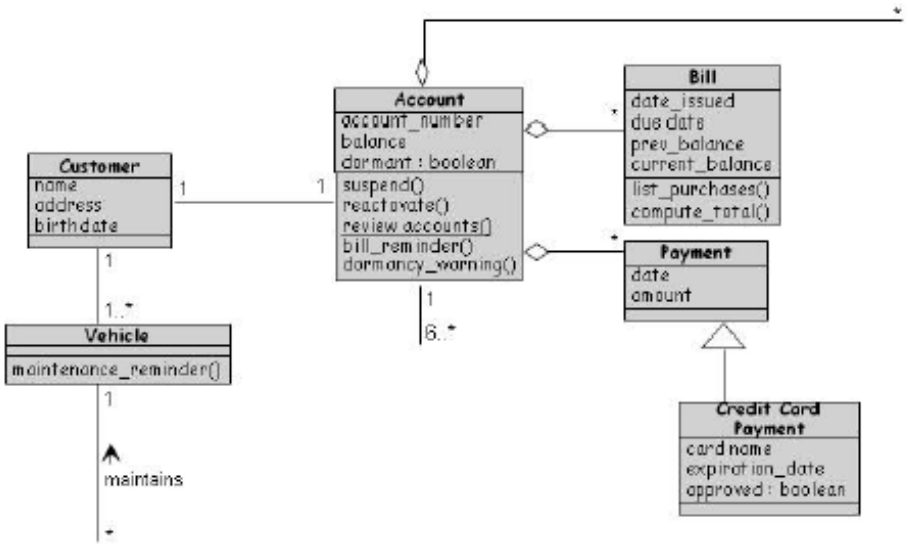


OO Measures



Metric	Min	Max
Number of Key Classes	21	21
Class Size	0	9
Number of Operations Overridden	0	1
Number of Operations Added	0	0
Specialization Index	0	1

Chidamber-Kemerer Metrics



Metric	Bill	Payment	Credit Card Payment	Account	Customer	Vehicle
Weighted Methods / Class	2	0	0	5	0	1
Number of Children	0	1	0	0	0	0
Depth of Inheritance Tree	0	0	1	0	0	0
Coupling Between Objects	1	2	1	5	2	2

Specialization Index (SIX)

SIX = NOO x DIT / TCM

Calculate the specialization index (SIX) for a set of classes given that Number of operations overridden by a subclass (NOO) is 3, the depth of inheritance tree (DIT) is 2, and total class methods (TCM) is 36.

SIX = 3 x 2 / 36
SIX = 6/36
SIX = 0.166

SIX = 30 x 7 / 36
SIX = 210/36
SIX = 5.83

Project Management
Numerical Problems

COCOMO: Effort, Time, Staff Size

E = a(KLOC)^b

Time = c(Effort)^d

Person required = Effort/ time

E = 3 (200)^1.12
E = 1133.117 person months

Time = 2.5 (Effort)^0.35
Time = 29.3 months

Person required = Effort/ time
P = 1133.1 / 29.3 = 38.7 persons

Effort, Time, Staff Size

- Size = 33,200 LOC, Productivity = 620 LOC/pm, Labour = 8000 $/Month, Compute effort and project cost, duration with 2 developers
  - Effort = Size / Productivity = LOC/P = 33,200/620 = 53.5 pm
  - Project Cost = Effort x Labour Cost = 53.5 pm x 8000 $/pm = 4,28,388 $
  - Duration = Effort / Resources = 53.5 pm / 2p = 26.75m

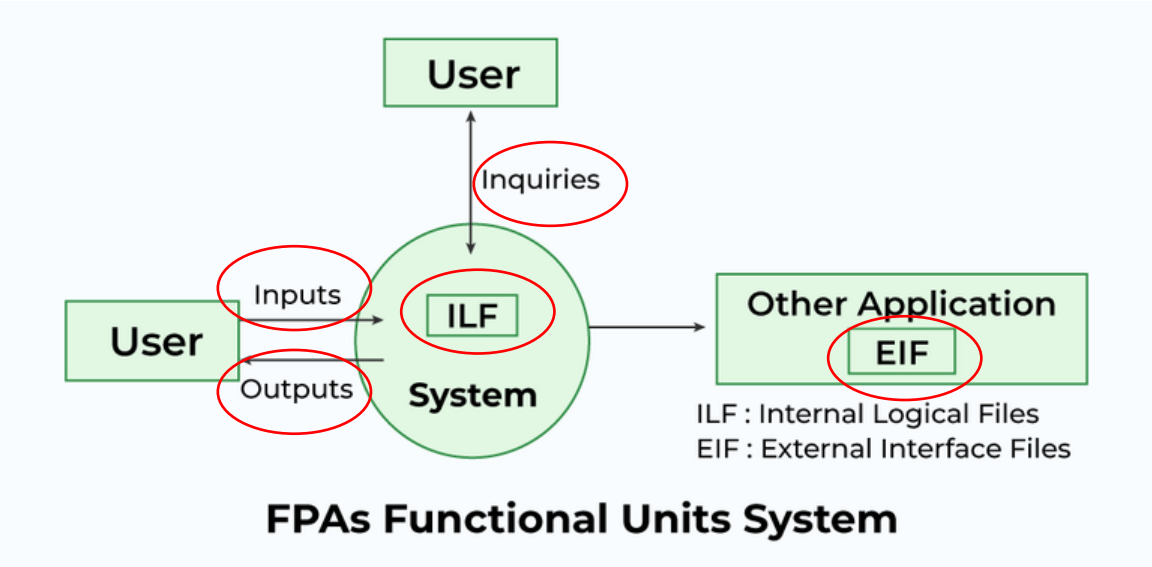
How does Project Management Begin?

1. Identify the project manager
2. Issue the Project Charter
    1. Authorises the PM to execute the project
    2. License to run the project (spend money, hire people, proceed with the project)
    3. Customer, Sponsor, PM, Client, End-Client

https://clickup.com/blog/project-charter-example/

Estimation of Size

- Function Point
- Feature Points
- Use Case Points



FPAs Functional Units System

$$Size\ in\ FP = FP \times (0.65 + 1\%\ of\ VAF)$$

Where FP = Inputs x Wi + Inquiries x Win + Outputs x Wo + ILF x Wilf + EIF x Wif

VAF = Value Adjustment Factor = $\sum_{i=1}^{14} VAFi \times Wi$

FP VAF

1.Data Communication	3	
2.Distributed data processing		0
3.Performance	1	
4.Heavily used configuration		0
5.Transaction rate	0	
6.Online data entry	2	
7.End user efficiency		1
8.Online update	0	
9.Complex processing	0	
10.Reusability	4	
11.Installation ease	0	
12.Operational ease		4
13.Multiple sites	0	
14.Facilitate change	0	

Estimation of Size

FP = 30 x 4 + 60 x 5 + 23 x 4 + 8 x 10 + 2 x 7
FP = 606

VAF = 4*0 + 4*3 + (14-4-4)*4 = 0 + 12 + 6*4 = 36

Size = FP x (0.65 + 1% of VAF)
Size = 606 x (0.65 + 0.01 x 36)
Size = 606 x (0.65 + 0.36)
Size = 606 x 1.01
Size = 612.06

Consider a software project with the following information domain characteristic for the calculation of function point metric: Number of external inputs (I) = 30, Number of external output (O) = 60, Number of external inquiries (E) = 23, Number of files (F) = 08, Number of external interfaces (N) = 02. It is given that the complexity weighting factors for I, O, E, F, and N are 4, 5, 4, 10, and 7, respectively. It is also given that, out of fourteen value adjustment factors that influence the development effort, four factors are not applicable, each of the other four factors has value 3, and each of the remaining factors has value 4. Compute the value of function point metric.

Estimation of Size

FP = 20 x 3 + 10 x 5 + 32 x 4 + 10 x 10 + 8 x 7
FP = 60 + 50 + 128 + 10 + 56
FP = 304

VAF = 5*0 + 4*4 + (14-5-4)*5 = 0 + 16 + 5*5 = 41

Size = FP x (0.65 + 1% of VAF)
Size = 304 x (0.65 + 0.01 x 41)
Size = 304 x (0.65 + 0.41)
Size = 304 x 1.06
Size = 322.24

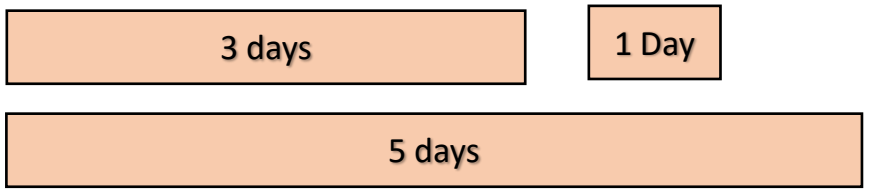
Consider a software project with the following information domain characteristic for the calculation of function point metric: Number of external inputs (I) = 20, Number of external output (O) = 10, Number of external inquiries (E) = 32, Number of files (F) = 10, Number of external interfaces (N) = 08. It is given that the complexity weighting factors for I, O, E, F, and N are 3, 5, 4, 10, and 7, respectively. It is also given that, out of fourteen value adjustment factors that influence the development effort, five factors are not applicable, each of the other four factors has value 4, and each of the remaining factors has value 5. Compute the value of function point metric.

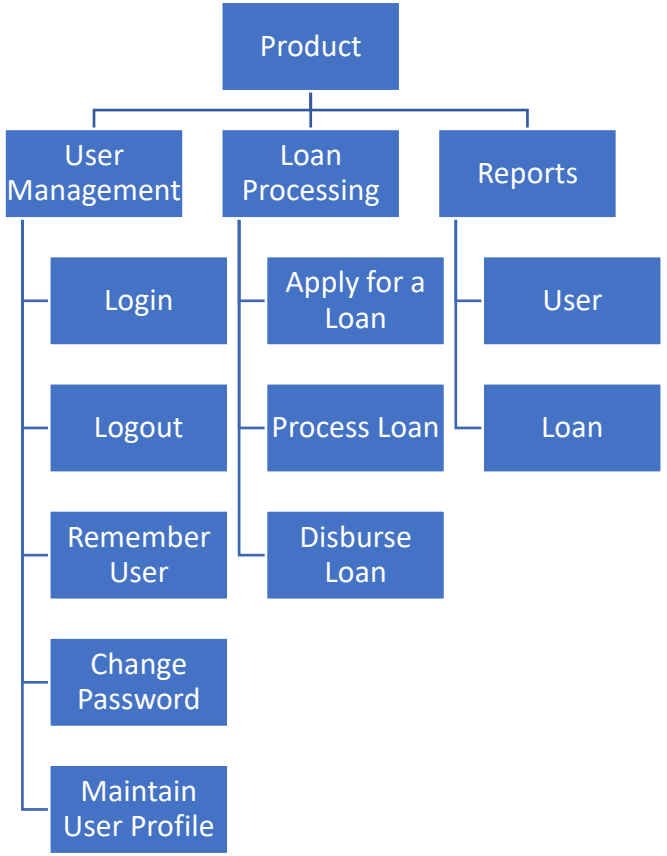
Breaking Down of Product into Tasks

Breakdown

Max Duration = 3 days

Work expands to time allocated (Parkinson's Law)



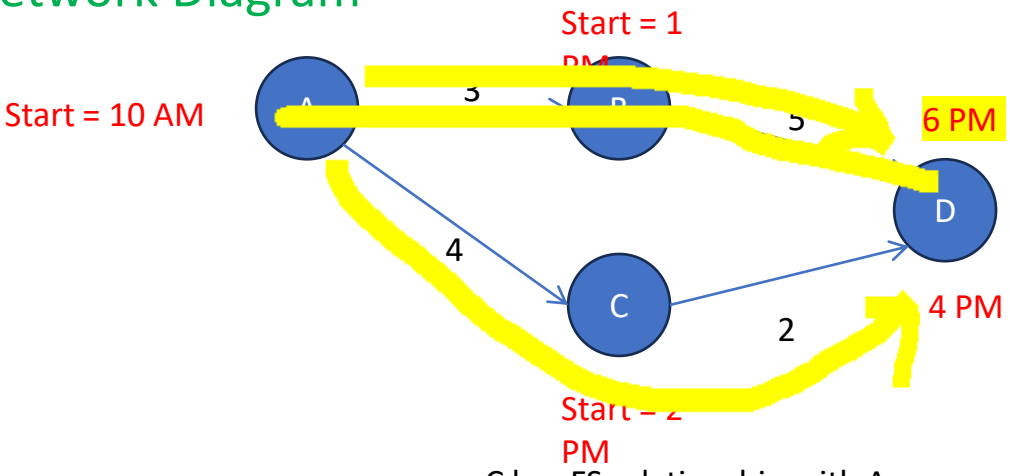


Resolve Task Dependencies

Tasks depend on other Tasks

Network Diagram

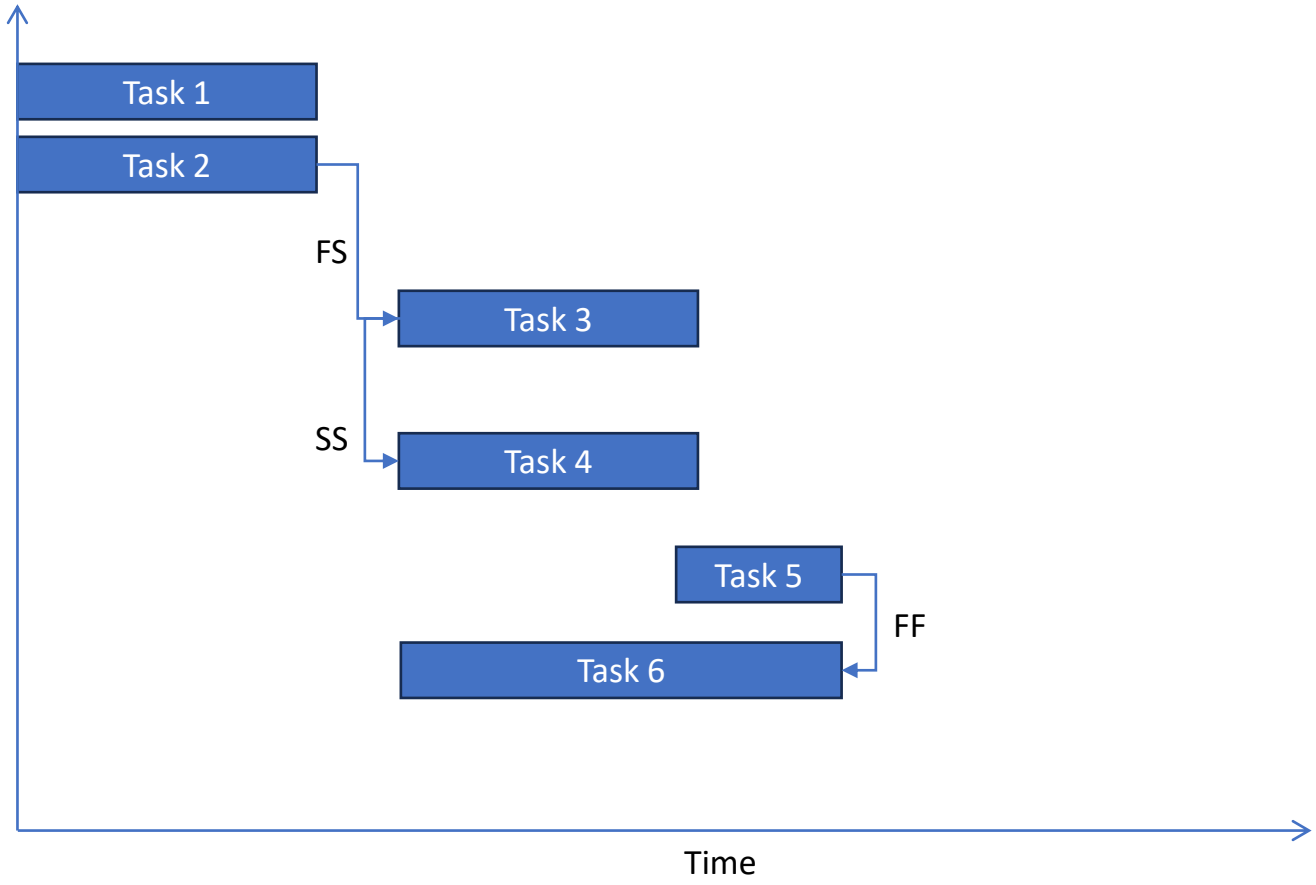
B has FS relationship with A
B can start only after A completes



D has FS relationship with both B and C
D can start only after both B and C completes

C has FS relationship with A
C can start only after A completes

Task Dependencies



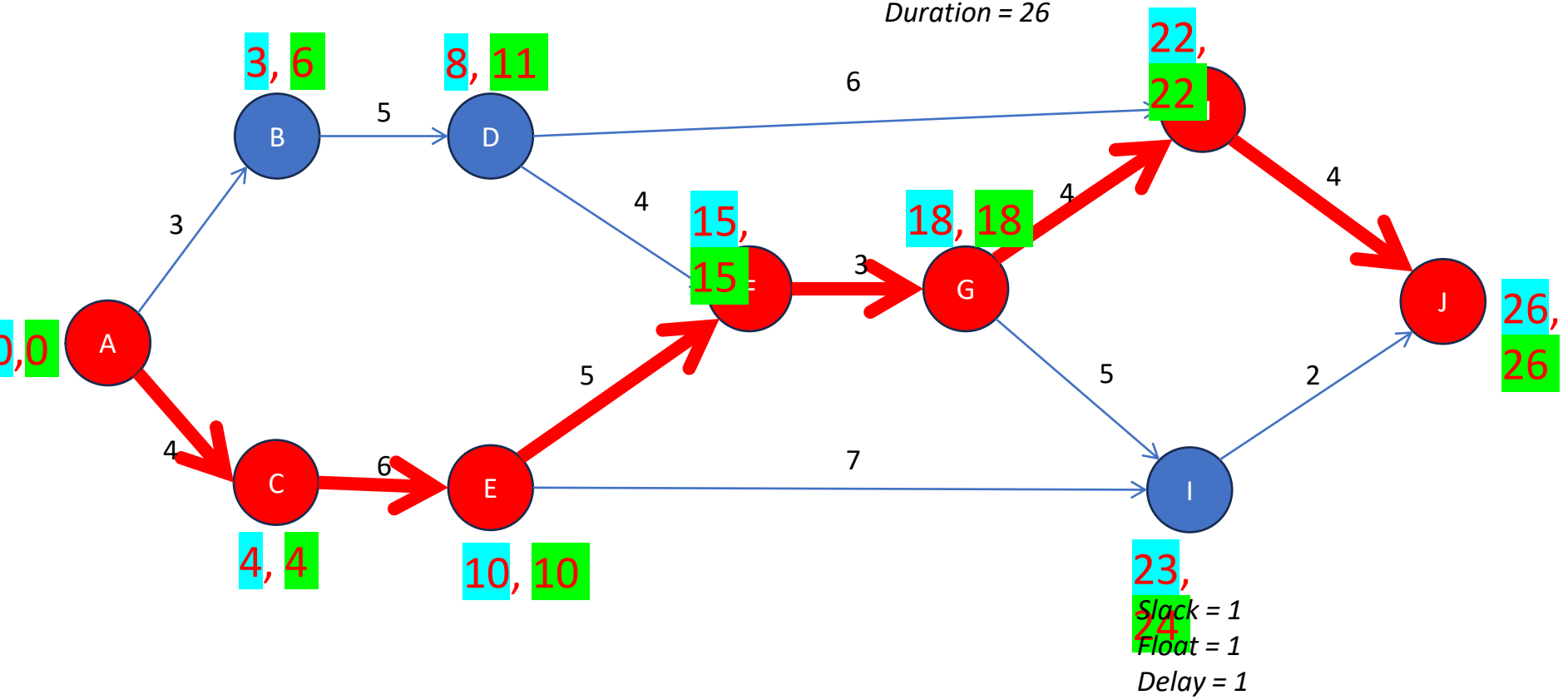
PERT/CPM: The Critical Path & Its Duration

Program Evaluation and Review Technique / Critical Path Method

Out of these 10 tasks, which tasks are on critical path?

A, C, E, F, G, H, and J

Duration = 26



Importance of Critical Path

Every task on critical path is critical

Critical Task has NO float! (Zero Float)

Can’t afford to delay the critical task

If any task on critical path is delayed --- then --- the project gets delayed!

Duration of critical path = duration of project

Duration of project = Duration of critical path

Agile: Calculation of Number of Sprints

In a project, each story is worth ten points. At the end of the sprint, your team has completed eight stories, but the ninth is only half complete. What is the velocity of the sprint? If the project includes 48 stories and the team performance remains the same, how many sprints are needed to complete the project?

Story points per story = 10

Sprint Velocity = No of stories developed

Velocity = 8 (ignore incomplete ones)

No of sprints = Size in Stories / Velocity

No of sprints = 48 / 8 = 6 sprints

Left over stories = 48 – 8 = 40

Left over sprints = 40 / 8 = 5 more sprints

Combined Risk Exposure

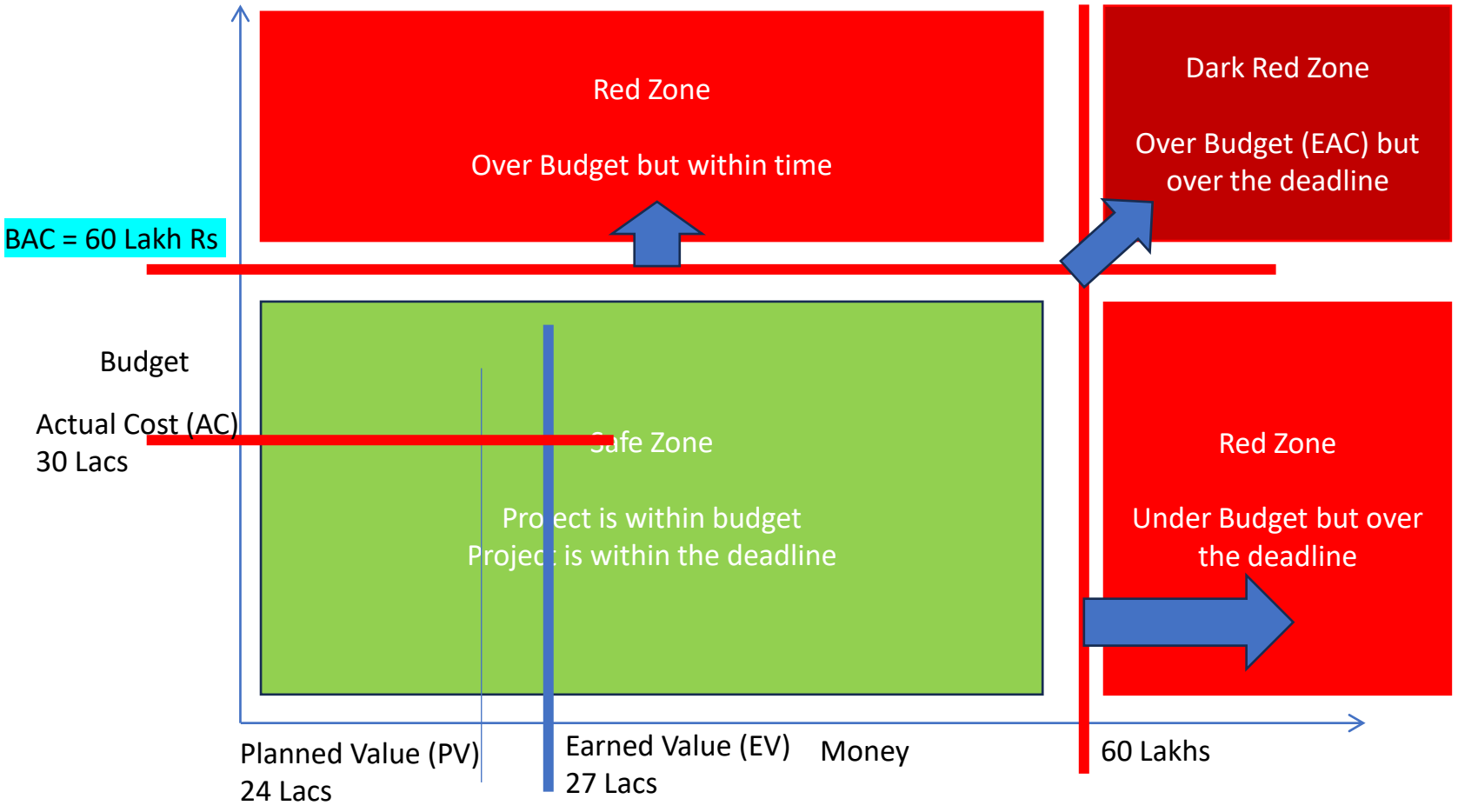
In a software project, three risks were identified.
(R1) 45% chance of losing 30 Lakh Rupees due to delayed delivery of reports module.
(R2) 15% chance of spending of 40 Lakh Rupees due to customer changing product specifications.
(R3) 30% chance of wasting 5 Lakhs due to delay in loading of customer supplied data.
Calculate the combined risk exposure of the project.

Combined Risk Exposure = $\sum_{i=1}^{n} Probabilityi \times Impacti$

CRE = (0.45 x 30) + (0.15 x 40) + (0.30 x 5)
CRE = 13.5 + 6 + 1.5
CRE = 21 Lakhs

The combined risk exposure of the project is 21 Lakh Rupees.

Earned Value Calculations - Funda



Earned Value Calculation Formulae

Earned Value	EV	Estimated, but given in the exam
Budget at Completion	BAC	Given
Planned Value	PV	Given
Actual cost	AC	Is estimated, but given in the exam
Schedule Perf. Index	SPI	EV/PV
Schedule Variance	SV	EV – PV
Cost Perf. Index	CPI	EV / AC
Cost Variance	CV	EV – AC
Estimate at Completion	EAC	BAC / CPI
Variance		EAC – BAC

Earned Value Calculation Example

Compute Estimate At Completion (EAC) and Variance At Completion (VAC) if both SPI and CPI influence the project work when given variables are Budget At Completion (BAC) = 35 Lakh Rupees, Earned Value (EV) = 27 Lakh Rupees, Planned Value (PV) = 24 Lakh Rupees, and Actual Cost (AC) = 30 Lakh Rupees. Also, identify whether the project is behind schedule and whether the project is within the budget.

BAC = 35 Lakhs
EV = 27 Lakhs
PV = 24 Lakhs
AC = 30 Lakhs

Schedule Variance	SV	EV – PV	27 – 24	= 3 Lakhs	(+ve number = good)
Cost Variance	CV	EV – AC	27 – 30	= -3 Lakhs	(-ve number = bad)
Schedule Perf. Index	SPI	EV / PV	27/24	= 1.125	(> 1 means ahead of schedule)
Cost Perf. Index	CPI	EV / AC	27/30	= 0.90	(< 1 means over budget)
Estimates at Completion		EAC	BAC / CPI	= 35 / 0.90 = 38.889 Lakhs (Over budget)	
Variance of Budget		VAC	EAC – BAC	= 38.889 – 35 = 3.889 Lakhs over budget	

If original project was supposed to end in 48 months, how long will it take now?

New duration = Old duration / SPI = 48 months / 1.125 = 42.667 months (completing early)

Duration variance = New duration – old duration = 48-42.667 = 5.333 months