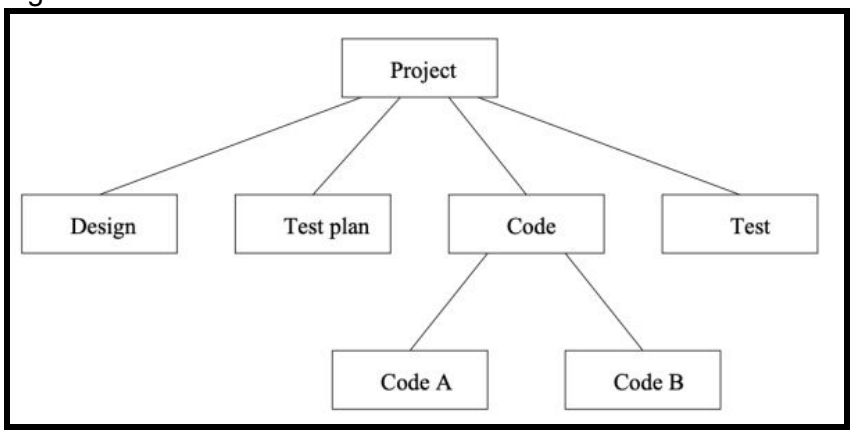
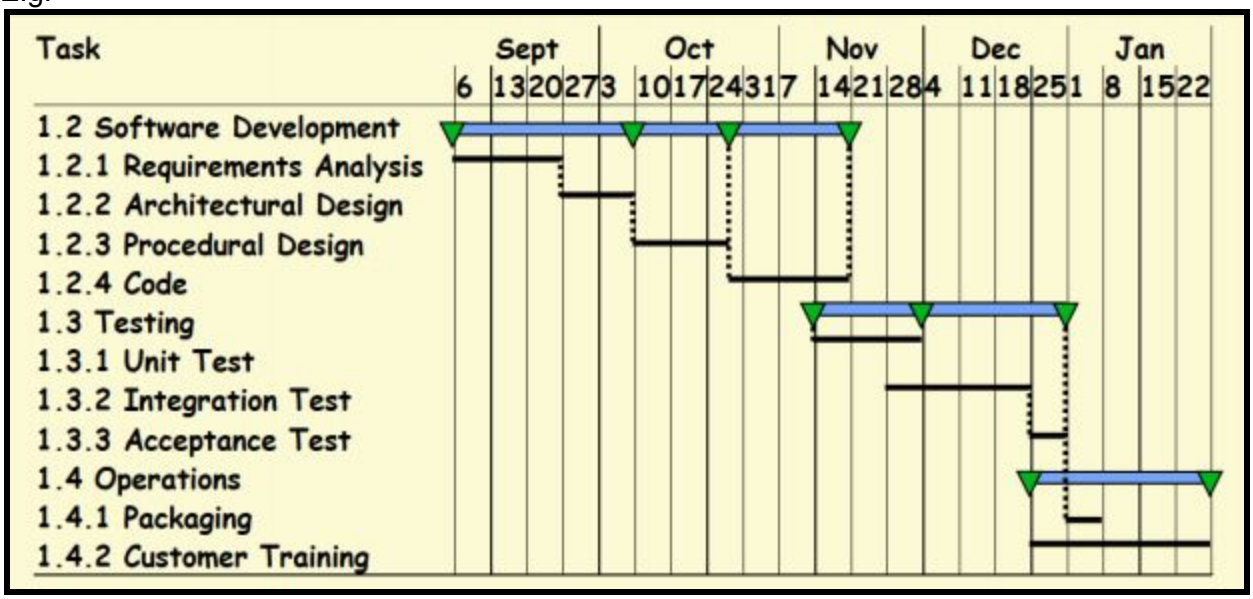
Project Management:

- **Key Principles of Management:**
- A manager can control 4 things:
    1. Resources - Can get more money, personnel, etc
    2. Time - Can vary the schedule and delay milestones
    3. Product - Can vary the amount of functionality
    4. Risk - Can decide which risks are acceptable
- To do this, a manager needs to keep track of the following items:
    - Effort - How much effort will be needed? How much effort has been spent?
    - Time - What is the expected schedule? How far are we deviating from it?
    - Size - How big is the planned system? How much have we built?
    - Defects - How many errors are we making? How many errors are we getting?
- Approach:
    - Understand the goals and objectives.
    - Understand the constraints.
    - Plan to meet the objectives within the constraints.
    - Monitor and adjust the plan.
    - Preserve a calm, productive, positive work environment.
- **Note:** You cannot control what you cannot measure.
- **Project Management Tools:**
- Work Breakdown Structure:
- A **work breakdown structure (WBS)** is a hierarchical decomposition of the total scope of work to be carried out by the project team to accomplish the project objectives and create the required deliverables.
- E.g.



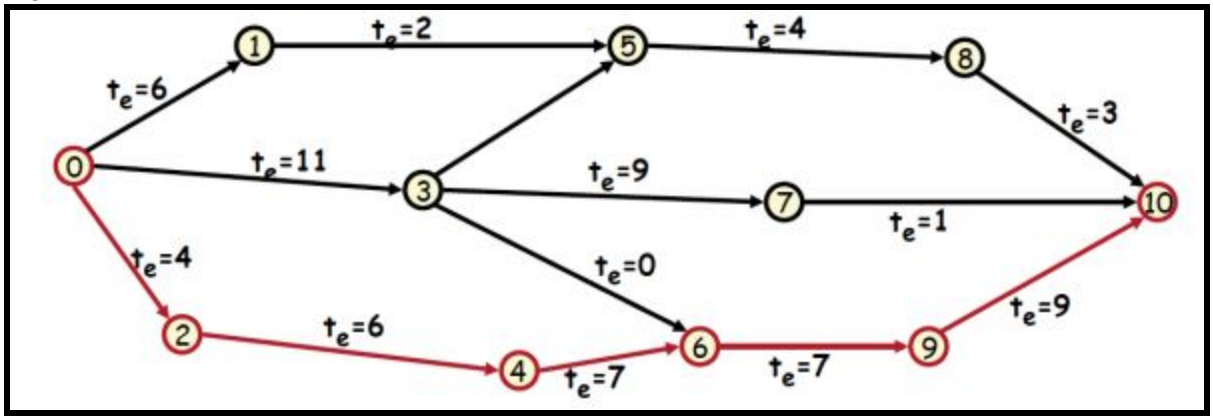
- Gantt Charts:
- A **Gantt chart** is a bar chart that provides a visual view of tasks scheduled over time. It is used for planning projects of all sizes, and it is a useful way of showing what work is scheduled to be done on a specific day. It can also help you view the start and end dates of a project in one simple chart.

- E.g.



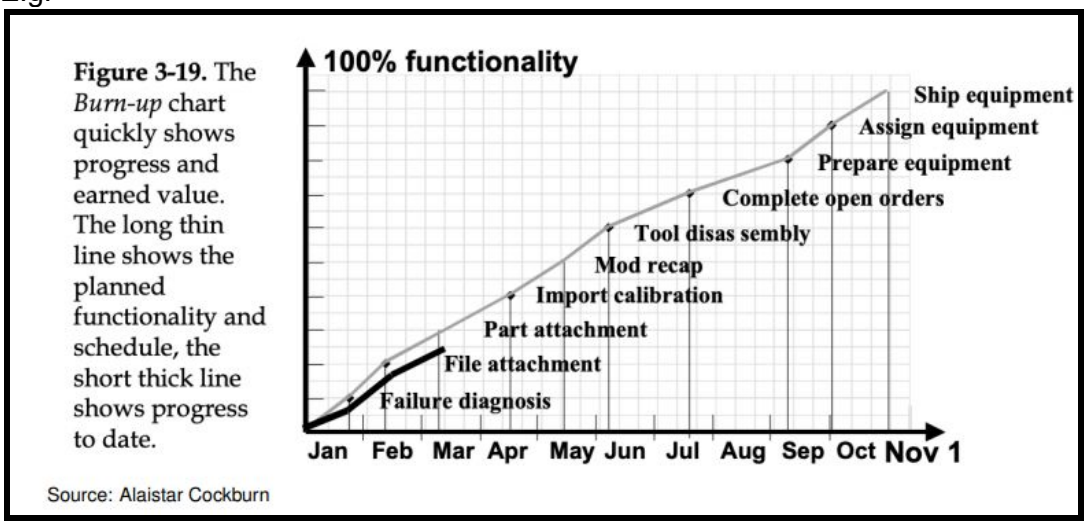
- Notations:
  - Bars show the duration of tasks.
  - Triangles show milestones.
  - Vertical lines show dependencies.
- PERT Charts:
- **PERT** stands for Program Evaluation and Review Technique. A PERT chart illustrates a project as a network diagram. The U.S. Navy created this tool in the 1950s as they developed the Polaris missile.
- Project managers create PERT charts to gauge the minimum time necessary to complete the project, analyze task connections, and assess project risk. PERT charts make it easy to visualize and organize complex projects illustrating the dependencies between each step in the project.
- PERT charts are best utilized by project managers at the beginning of a project to ensure that it is accurately scoped. This tool gives users a birds-eye view of the entire project before it's started to avoid potential bottlenecks. While PERT charts can be used during the project's implementation to track progress, they are not flexible enough for teams to adapt them to small changes when team members are confronted with roadblocks.
- Advantages of using PERT charts
  - A PERT chart allows managers to evaluate the time and resources necessary to manage a project. This evaluation includes the ability to track required assets during any stage of production in the course of the entire project.
  - PERT analysis incorporates data and information from multiple departments. This combining of information encourages department responsibility and it identifies all responsible parties across the organization. It also improves communication during the project and it allows an organization to commit to projects that are relevant to its strategic positioning.
  - Finally, PERT charts are useful for what-if analyses. Understanding the possibilities concerning the flow of project resources and milestones allows management to achieve the most efficient and useful project path.

- Disadvantages of using PERT charts:
    - The use of a PERT chart is highly subjective and its success depends on the management's experience. These charts can include unreliable data or unreasonable estimates for cost or time for this reason.
    - PERT charts are deadline-focused and they might not fully communicate the financial positioning of a project. Because a PERT chart is labor-intensive, the establishment and maintenance of the information require additional time and resources. Continual review of the information provided, as well as the prospective positioning of the project, is required for a PERT chart to be valuable.
- **Critical path** is the sequential activities from start to the end of a user story. Although many user stories have only one critical path, some may have more than one critical paths depending on the flow logic used in the user story implementation.
- A critical path is determined by identifying the longest stretch of dependent activities and measuring the time required to complete them from start to finish.
- If there is a delay in any of the activities under the critical path, there will be a delay of the user story delivery.
- Key steps in critical path method:
    1. **Activity specification:**
    - Break down a user story in a list of activities.
    2. **Activity sequence establishment:**
    - Need to ask three questions for each task of your list.
        1. Which tasks should take place before this task happens.
        2. Which tasks should be completed at the same time as this task.
        3. Which tasks should happen immediately after this task.
    3. **Network diagram:**
    - Once the activity sequence is correctly identified, the network diagram can be drawn.
    4. **Identify critical path:**
    - The critical path is the longest path of the network diagram. If an activity of this path is delayed, the user story will be delayed.
- Tasks on the critical path have to start as early as possible or else the whole project will be delayed. However tasks not on the critical path have some flexibility on when they are started. This flexibility is called the **slack time**.
- E.g.

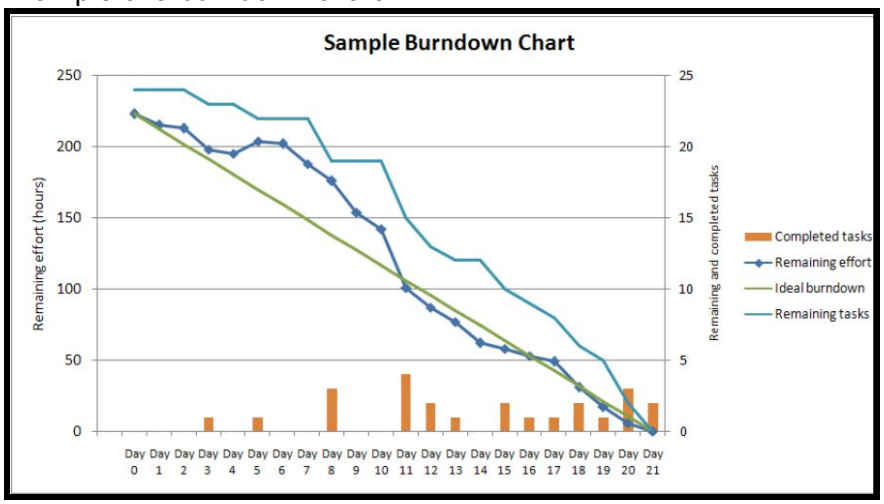


- Notation:
    - Nodes indicate milestones.
    - Edges indicate dependencies and are labelled with the time to complete it.

- Burn-up Charts:
- A **burnup chart** is a tool used to track how much work has been completed, and show the total amount of work for a project or iteration. Typically, in a burnup chart, the outstanding work is often on the vertical axis, with time along the horizontal. It is useful for predicting when all of the work will be completed.
- Burnup charts and burndown charts are quite similar and display much of the same information. Burndown charts are simple and easy for project members and clients to understand. A line representing the remaining project work slowly decreases and approaches zero over time. However, this type of chart doesn't show clearly the effects of scope change on a project. If a client adds work mid-project the scope change would appear as negative progress by the development team on a burndown chart.
- In contrast, scope changes are immediately evident on burnup charts. When new work is added the total work line will clearly show the increase in scope and total work.
- E.g.



- Burn-down Charts:
- Example of a burndown chart:

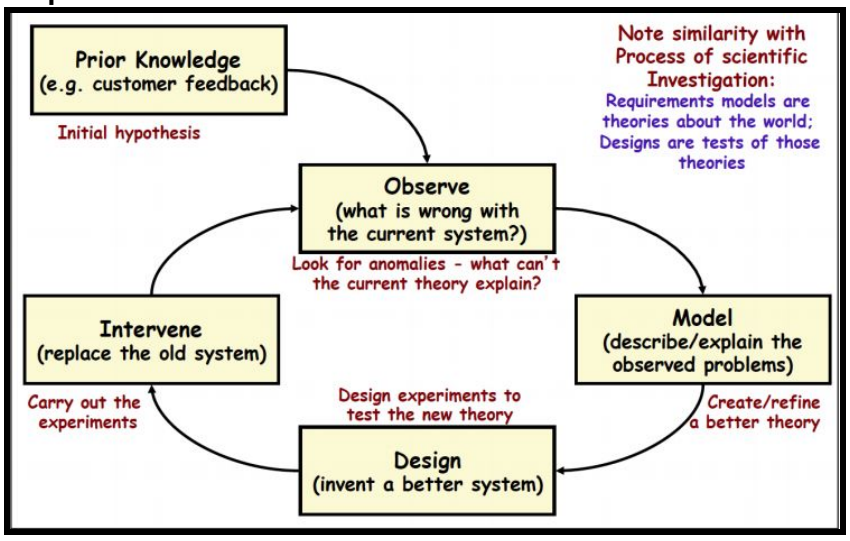


- Indicates how much work has been done in terms of how many user stories have been completed and when.
- Burndown charts are on Jira.

- On the beginning of the sprint, on the vertical axis, is the number of user stories you'd like to implement.
- At the end of the sprint, the number of user stories should be decreased to 0. That means all the stories have been implemented.
- A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum.
- Typically, in a burndown chart, the outstanding work is often on the vertical axis, with time along the horizontal. It is useful for predicting when all of the work will be completed.
- <u>Meetings as a management tool:</u>
- Meetings are expensive, so don't waste people's time and the company's money with unnecessary meetings. However, meetings are necessary for communications.
- Do's/Don'ts for meetings:
    - Do announce details in advance.
      I.e. The purpose of the meeting, the duration of the meeting, who should join, etc.
    - Do lay out a clear and concise agenda for the meeting.
    - Do identify a person who will lead the meeting.
    - Do identify a person who will take notes for the meeting.
    - Do not waste people's time by showing up unprepared (especially if you're in charge).
    - Do not let discussion get sidetracked.
    - Do not let one one or two people dominate the meeting.

<u>Requirement Analysis:</u>

- **Requirements as Theories:**



- **Starting point:**
- Given a vague request for a new feature from users of your software:
    1. Identify the problem:
    - Find out what the goal is.
    2. Scope the problem:
    - Find out what new functionalities are needed.
    3. Identify solution scenarios:
    - Find out how users will interact with the software to solve the problem.
    4. Map onto the architecture:
    - Find out how the new functionalities will be met.

- Given a problem, requirements analysts must identify the following:
    - Which problem(s) need to be solved?
    - Where are the problem(s)?
    - Whose problem is it?
    - Why does it need solving?
    - When does it need solving?
    - What might prevent us from solving it?
    - How might a software system help?
- **Terminology:**
- **Application Domain:** Things the software cannot observe directly. Includes domain properties and requirements.
- **Domain Properties:** Things in the application domain that are true, whether or not we ever build the proposed system.
- **Requirements:** Things in the application domain that we wish to be made true, by delivering the proposed system.
- **Software Domain:** Things private to the software. Includes computers and programs.
- **Specification:** A description of the behaviours the program must have in order to meet the requirements.

- **Verification:** The program running on a particular computer satisfies the specification and the specification, in the context of the given domain properties, satisfies the requirement.
- **Validation:** We discovered all the important requirements and properly understood the relevant domain properties.
- **Observations:**
- Analysis isn't necessarily a sequential process. Rewriting the problem statement can be useful at any stage of development.
- The problem statement will be imperfect. Models are approximations, not the actual thing. They will contain some inaccuracies and omit some information. We need to assess the risk that these will cause serious problems.
- Perfecting a specification may not always be cost effective.
- The problem statement should never be treated as fixed. Change is inevitable and therefore must be planned for.
- **Stakeholders:**
- **Stakeholder analysis** is identifying all the people who must be consulted during information acquisition.
- Type of stakeholders:
    - Users - Concerned with the features and functionalities of the new system.
    - Customers - Wants to get the best value for money invested.
    - Business analysts/Marketing team - Wants to make sure that we're doing better than the competition.
    - Training and user support staff - Want to make sure the new system is usable and manageable.

- Technical authors - Will prepare user manuals and other documentation for the new system.
- System analysts - Want to get the requirements right.
- Designer - Want to build a perfect system or reuse existing code.
- Project manager - Wants to complete the project on time, on budget and with all objectives met.

Goals:

- A **goal** is a stakeholder objective for the system.
- A **goal model** is a hierarchy of goals that relates the high-level goals to low-level system requirements.
- **Hard goals** describe the functions the system will perform.
  E.g. The system collects timetables from users.
- **Soft goals** describe the desired system qualities. Soft goals cannot be fully satisfied.
  E.g. The system should be reliable.
  E.g. The system should be high quality.
- Goal elaboration:
  - "Why" questions explore higher goals (context).
  - "How" questions explore lower goals (operations).
  - "How else" questions explore alternatives.
- Relationships between goals:
  - A goal helps another goal. (+)
  - A goal hurts another goal. (-)
  - A goal makes another goal. (++)
  - A goal breaks another goal. (--)
- Approach for identifying stakeholder goals:
  - Focus on why a system is required.
  - Express the "why" as a set of stakeholder goals.
  - Use goal refinement to arrive at specific requirements.
  - Document, organize and classify goals.
  - Refine, elaborate and operationalize goals.

Use Case Diagrams:

- **Introduction:**
- A **use case diagram** is the primary form of system/software requirements for a new software program underdeveloped.
- Use cases specify the expected behavior (what), and not the exact method of making it happen (how).
- A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.
- Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified.
- A use case:
  - Specifies the behavior of a system or some subset of a system.
  - Is a set of scenarios tied together by a common user goal.
  - Does not indicate how the specified behavior is implemented, only what the behavior is.
  - Performs a service for some user of the system.
  - A user of the system is known as an **actor**.

    - During the analysis phase, facilitates communication between the customer, users of the system and the developers of the system.
- A use case represents a functional requirement of the system. A requirement:
    - Is a design feature, property, or behavior of a system.
    - States what needs to be done, but not how it is to be done.
    - Is a contract between the customer and the developer.
    - Can be expressed in various forms, including use cases.
- In brief, the purposes of use case diagrams are as follows:
    - Used to gather the requirements of a system.
    - Used to get an outside view of a system.
    - Identify the external and internal factors influencing the system.
    - Show the interaction among the requirements of the **actors**.
- **Actors** can be a human user, some internal applications, or may be some external applications. It is a user of the system.
- An actor:
    - Is a role that the user plays with respect to the system.
    - Is associated with one or more use cases.
    - Does not have to be human.
    - Is a user of the system.
    - Is most typically represented as a stick figure of a person labeled with its role name. Note that role names should be nouns.
    - May exist in a generalization relationship with other actors in the same way as classes may maintain a generalization relationship with other classes.
- **Note:** Use cases only show the relationships between actors, the systems and use cases and does not show the order in which the steps are performed to achieve the goals of each use case.
- Use cases are a technique for capturing the functional requirements of a system. Use cases work by describing the typical interactions between the users of a system and the system itself, providing a narrative of how the system is used.