SOFTWARE PROJECT MANAGEMENT

1. Introduction

- The "*software crisis*" of the 1960s and 1970s was so called because of a string of high profile software project failures: over budget, overdue, etc.
- The crisis arose in part because the greater power available in computers meant that larger software projects were tackled with techniques developed on much smaller projects.
- Techniques were needed for *software project management.*
- Good project management cannot guarantee success, but poor management on significant projects always leads to failure.

- Software project has properties that make them very different to other kinds of

- *The product is intangible.*

  Its hard to claim a bridge is 90% complete if there is not 90% of the bridge there.

  Its easy to claim that a software project is 90% complete, even if there are no visible outcomes.

- *We don't have much experience*

  How to engineer large scale software projects.

- *Large software projects are often “bespoke”.*

  Most large software systems are one-off, with experience gained in one project being of little help in other.

- *The technology changes very quickly*

Activities in software project management

- Project planning
- Project scheduling
- Risk management
- Managing people

2. Project planning

- The biggest problem that afflicts software developing is that of underestimating the resources required for a project

- A realistic project plan is essential to:
  - Gain understanding of the resources required
  - How these resources should be applied

Types of project plan

Plan	Description
Software development plan	The central plan, which describes how the system will be developed.
Quality assurance plan	Specifies the quality procedures and standards to be used.
Validation plan	Describes how the client will validate the system that has been developed.
Configuration management plan	Defines how the system will be configured and installed.
Maintenance plan	Defines how the system maintained
Staff development plan	Describes how the participants will be developed

2.1 The software development plan

- This is usually what is meant by a project plan
- Specifies the order of work to be carried out, resources, resources, responsibilities and so on.
- Varies from small and relatively informal to large and very formal.
- Developing a project plan is as important as properly designing code:

On the basis of a project plan, contracts will be signed and careers made or broken.

- Important not to:
  - Overestimate your team’s ability.
  - Simply tell clients what they want to hear
  - Be pressured by developers

2.2 Structure of development plan

1. *Introduction*

   Brief intro to project– references to requirement phase

2. *Project organization*

   Intro to organizations, people, and their roles

3. *Risk analysis*

   What are the key risks to the project

4. *Hardware and software resources*

   What hardware and software resources will be required for the project and when?

5. Work breakdown

the project divided into activities, milestones, deliverables, dependencies between tasks etc.

6. Project schedule

actual time required– allocation of dates

7. Reporting and project measurement

Mechanism to monitor progresses

2.3 Work breakdown

- There are many ways of breaking down the activities in a project, but the most usual is into:
  - Work packages
  - Tasks
  - Deliverables
  - milestones

- A *workpackage* is a large, logically distinct section of work:
  - Typically at least 12 months duration
  - May include multiple concurrent activities
  - Independent of other activities
  - But may depend on, or feed into other activities
  - Typically allocated to a single team
- A *task* is typically a much smaller piece of work:

A part of workpackage

  - Typically 3-6 person months effort;
  - May be dependent on other concurrent activities
  - Typically allocated to a single person

- A *deliverable* is an output of the project that can meaningfully be assessed.

Examples:

  - a report (e g., requirements spec);
  - Code (e g., alpha tested product)

  Deliverables are indicators (but only indicators) of progress

- A *milestone* is a point at which progress on the project may be assessed.

  Typically a major turning point in the project.

  EXAMPLES:

  - Delivery of requirements spec;
  - Delivery of alpha tested code.

- Usually..
  - Work packages are numbered WP1,WP2 etc
  - Tasks are numbered TI.1, T2.1 etc, the first number is the number of the workpackage;

    The second is a sequence number.
  - Deliverables are numbered D1.1, D1.2, etc.
  - Milestones are numbered M1.1, M1.2, etc.

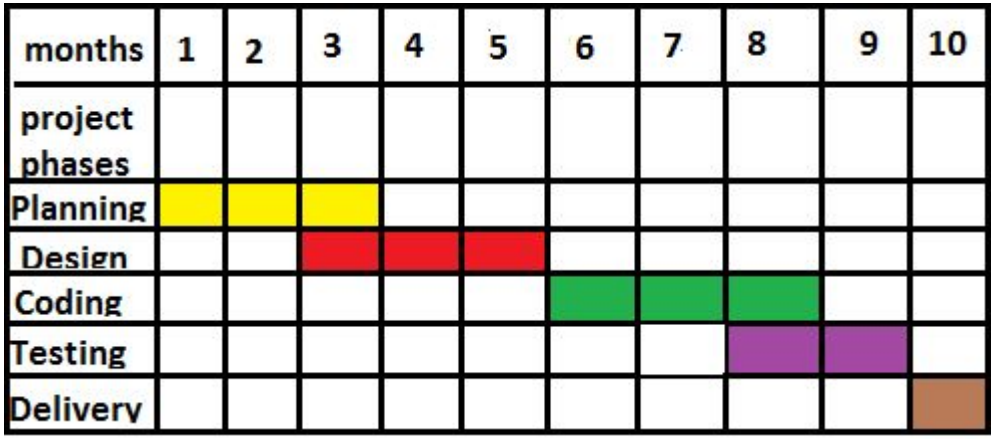
- For each work package and task, it is usual to document:
  - Brief description
  - Earlier start date
  - Earliest end date
  - Total person months effort
  - Pre- requisite WPs or tasks
  - Dependent WPS or tasks
  - Who is responsible

2.4 Critical paths

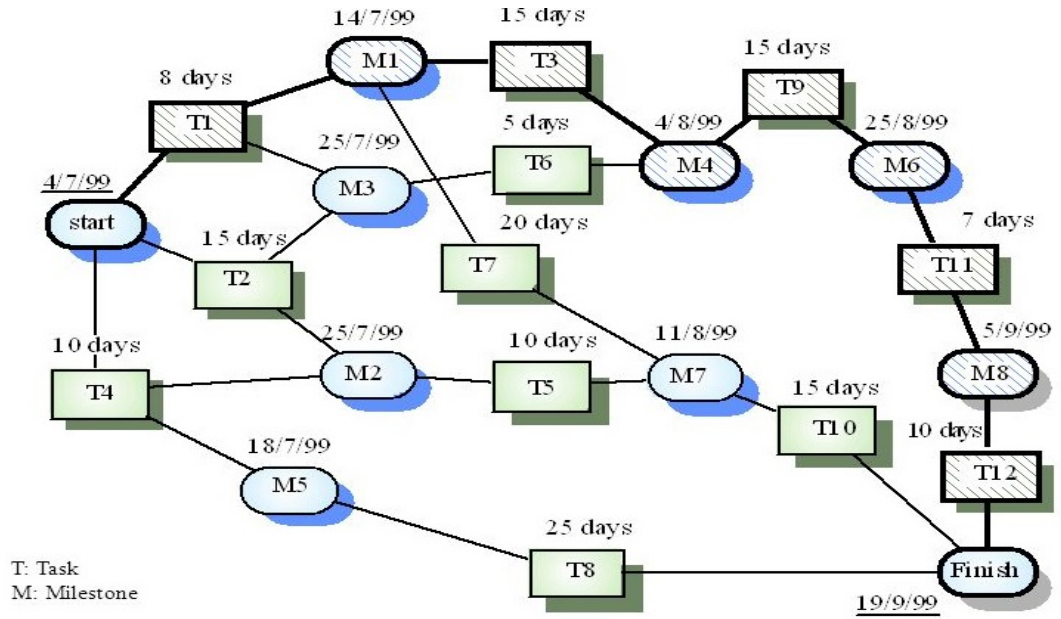
- The pre- requisites and dependencies of WPs and tasks determine a *critical path:* the sequence of dependencies in the project.
- The critical path is the sequence of activities that takes the longest tie to complete.
- Any delay to an activity in the critical path *will* cause delays to the overall project.
- Delays to activities not on the critical path need not necessary cause overall delays.

3. Gantt charts and Activity networks

- Gantt charts are a kind of bar chart:
  - Time plotted on *x* axis
  - Bars on *y* axis for each activity



- An *activity network* is a labeled graph, with:
  - Nodes corresponding to activities
  - Arcs labeled with estimated times
  - Activities are linked if there is a dependency between them



4. Risks

- When planning a project, it is critically important to know what the key risks are, and is possible plan for them:
  - *Staff turnover*
  - *Management change*
  - *Hardware unavailability*
  - *Requirements change*
  - *Specification delays*
  - *Sixe underestimate*
  - *Technology change*
  - *Product completion*

5. Quality assurance

- Many organizations make use of a quality assurance plan, which sets out standards to be maintained during project development.
  - Documentation standards:
    - What documents
    - Format and content
  - Coding standards
    - Class/ method/ variable naming conventions
    - Comment standards (eg: javadoc)
    - Testing conventions