

4. Software Project Management

Subject/Topic/Focus:

- Management of Systems Engineering Projects

Summary:

- Influences on projects and potential risks
- Software systems engineering and project management
- Activity network and activity bar chart; PERT chart and Gantt chart
- Staff allocation and man-month
- Process improvements

Literature:

- [Sommerville 1996]
- [Brooks 1972]
- [Fowler 1997]
- Grady Booch: **Best of Booch**. SIGS 1996.

Systems Engineering

“... is the selective application of scientific and engineering efforts to

- transform an operational need into a description of the system configuration which best satisfies the operational need according to the measures of effectiveness;
- integrate related technical parameters and ensure compatibility of all physical, functional, and technical program interfaces in a manner which optimizes the total system definition and design;
- integrate the efforts of all engineering disciplines and specialties into the total engineering effort.”

[USALMC Army Field Manual: “Systems Engineering”.
Training Support Center, Ft. Eustis, 1978]

Project Management:

- Balances cost, schedule, quality, and functionality objectives.
- Systems engineering provides engineering viewpoint in business decisions.

Systems Engineering: Example

Denver International Airport, built in the early 1990s

Opening delayed for 25 months → \$375,000,000 loss
Costs per day of delay: \$500,000 →

Main reason for delay:

- faulty system's software and hardware of baggage system using telecars

Errors occurred although

- the prime-contractor was well experienced,
- redundancies were built into the system and
- the components were tested.

But:

- The size and complexity of the system requirements were unique.

[Linda Geppert: „Faults and Failures“. IEEE Spectrum, August 1994]

Influences on Projects

- Maturity of **technology**:
 - opportunities ... risks
- **Business** influence:
 - prototypical ... mission-critical
- Implementation **culture**:
 - Cobol ... LISP ... Java
- **Domain**:
 - Nuclear plant control ... management information system ... video game
- Experience of **team** members:
 - rookie ... freak ... expert
- **Scope**:
 - isolated application ... enterprise-wide information system

OO Project Maturity

Distinguish

- **isolated** application development of independent applications from
- **enterprise-wide** development of application families.

Project maturity of object-orientation shows in terms of

- **architecture**
 - consolidate repeating (consistent) patterns
 - institutionalize a chief architect
- **process**
 - formalize, validate, re-evaluate and re-design the process continuously
 - refine the architecture (patterns) regularly estimating costs & gains
- **documentation**
 - systems have to endure the lifetimes of their creators
 - standardize, deliver and review (architecture) documentation [Booch]

OOAD Project Experience

Do not underestimate the importance of **failure** in object-oriented development.

[Booch]

The **first projects** using a new technology, method, language, ... will **fail** - so their goal is to **gain experience**. Keep them small - a company that depends on the production results of these projects is doomed.

A **customer** who depends on his business processes, who cannot afford even minutes of down-time (consider bank or stock market transactions), does not approve the elegance, modernity, reusability, extensibility, ... of your solution, but its **reliability**.

The **technology** (method, language, ...) used in a project does not determine the result. It is determined by the **team** who drives the project, the experience of the team members and their cooperative work.

To remain nimble in the marketplace, you have to know when to **establish** corporate **guidelines** and when to **break** them.

[Booch]

Risks Categories

- **Requirements** risks Build the right system.
 - What are the requirements of the system?
 - Avoid misunderstanding the priorities and building the wrong system, one that does not do what the customer wants it to do.
- **Technological** risks Build the system right.
 - Can you handle OO-design technology?
 - How well does this technology work?
"You have been told to use Java and the Web. Can you actually deliver the functions that users need through a Web browser connected to a database?"
- **Skill** risks Choose the right system builders.
 - Can you get the staff and expertise you need?
- **Political** risks Consider who wants the system (not) to be built.
 - Are there political forces that can get in the way and seriously affect the project ?

Risks are identified during **elaboration!**

[Fowler]

Technological Risks

The most important thing to do in addressing technological risks is to **build prototypes** that try out the pieces of technology you are thinking of using.

- Build a simple part of an early version of the domain model.
- Try several tools. See which one is the best suited for the job and which ones are the easiest to use.
You want to use C++ and a relational database:
 - Get the C++ compiler and other tools.
 - Build the database and connect it to the C++ code using the different tools.

The biggest technological risks are inherent in how the **components** of a design **fit together**, rather than present in the components themselves.

Technological Risks: Hints

- **Focus** on elements which appear **difficult to change** later.
- Try to do your design in a **modular** way to allow to **exchange** single parts.
Ask yourself these questions:
 - What will happen if a piece of technology does not work?
 - What if we cannot connect two pieces of the puzzle?
 - What is the likelihood of something going wrong? How would we cope if that happens?
- Use UML tools to investigate for “purple worms” in your design.
 - **Class** diagrams show the overall structure of your system.
 - **Interaction** diagrams exemplify how components communicate.
 - **Package** diagrams deliver a high-level picture of the components.

Skills Risks

- Main problem of OO-projects is the **lack of training**.
- If you worry about the **costs** of training you will pay every penny, as the project takes longer!
- A good way to improve your OO-skills is through **mentoring**.
 - Work together with an experienced developer.
 - Obtain tips and hints from a mentor.
- Use the expertise and the skills of a mentor or an experienced developer in the **early** phases of your project. As time goes on you become more capable and the mentor does more **reviewing** than developing.

Project Management

Management techniques derived from small-scale projects do not scale up to large systems development.

Projects deficiencies: Software

[Sommerville]

- is delivered late,
- is unreliable,
- costs several times the original estimates,
- exhibits poor performance,
- does not meet its requirements.

Note: These are technology issues, but caused by management!

Challenges for software managers in contrast to other engineering disciplines:

- The product is intangible. Progress cannot be reviewed directly.
- There is no standard process for software development.
- Large software projects are often “one-of” projects. Experience may not be transferred easily.

Software Manager Responsibilities

Proposal writing

- objectives
- justification
- rough working plan
- cost and schedule estimates

Project costing

- estimate
- book-keeping & re-estimation

Project planning and scheduling

- identify activities, milestones, deliverables
- plan development guide
- estimate resources required

Project monitoring and reviews

- keep track of progress
- compare actual to planned progress
- informal on daily basis
- formal at scheduled reviews
- adapt to objective changes

Personnel selection and evaluation

- weigh experience vs. cost vs. availability
- consider skill development (learning process)

Report writing and presentation

- concise, coherent, abstract
- for clients & contractors

Software Development Plan

supplemented by further plans, see next slide

Introduction

- objectives & constraints (budget, time, staff, ...)

Project organization

- team organization & people involved & roles assigned

Risk Analysis

- identification and likelihood of risks & risk reduction strategies

Hardware and software resource requirements

- including prices and delivery schedule

Work breakdown

- activities & milestones & deliverables of activities

Project schedule

- dependencies between activities & estimated milestone dates & allocation of people to activities

Monitoring and reporting mechanisms

- including management report structure & delivery dates

Auxiliary Plan Types

Quality plan

describes project quality procedures & standards

Validation plan

describes system validation approach, resources & schedule

Configuration management plan

describes configuration management procedures and structures

Maintenance plan

predicts system maintenance requirements, costs and effort

Staff development plan

describes how skills and experience of project team members will be developed

Planning takes most management time!

Project Planning Procedure

Establish the project constraints
 Make initial assessments of the project parameters
 Define project milestones and deliverables
while project has not been completed or cancelled **loop**
 Draw up project schedule
 Initiate activities according to schedule
 Wait (for a while, usually 2-3 weeks) + exceptions!
 Review project progress
 Revise estimates of project parameters
 Update the project schedule
 Re-negotiate project constraints and deliverables
 if (problems arise) **then**
 Initiate technical review and possible revision
 end if
end loop

Activity Organization

Managers need information, provided by documents, to control the process.

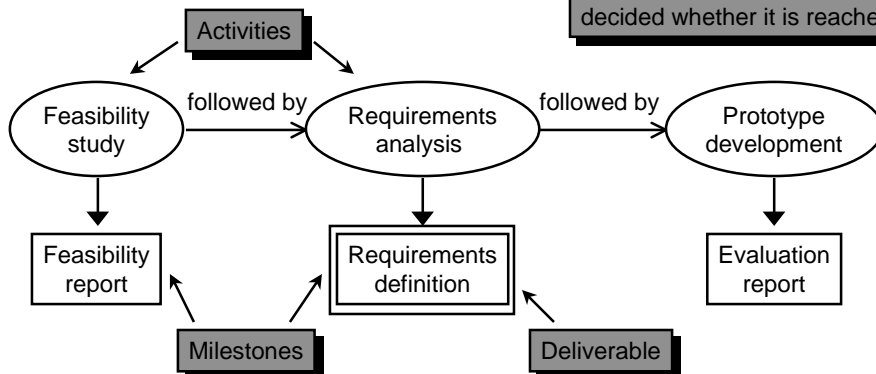
Milestone: End-point of **activity**.

- short formal progress report

Coding 80 % complete.

Deliverable: milestone delivered to customer.

This is **not** a milestone because it cannot be decided whether it is reached.



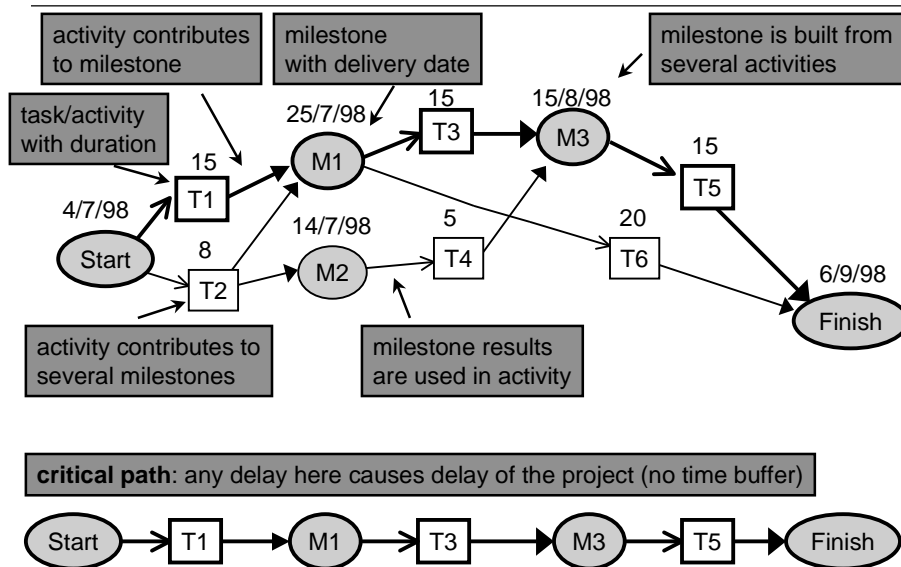
Task Duration and Dependencies

Task/Activity	Duration (days)	depends on
T1	15	
T2	8	
T3	15	T1, T2
T4	5	T2
T5	15	T3, T4
T6	20	T1, T2

Typical duration of tasks/activities: 1-10 weeks

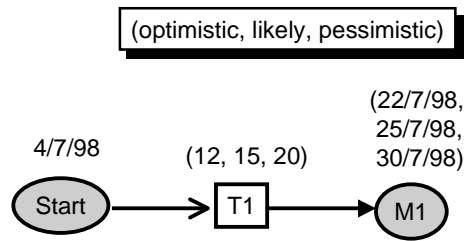
- finer grained (days): increases time needed for reviewing and re-scheduling
- coarser grained (months): decreases control and delay detection

Activity Network / PERT Chart

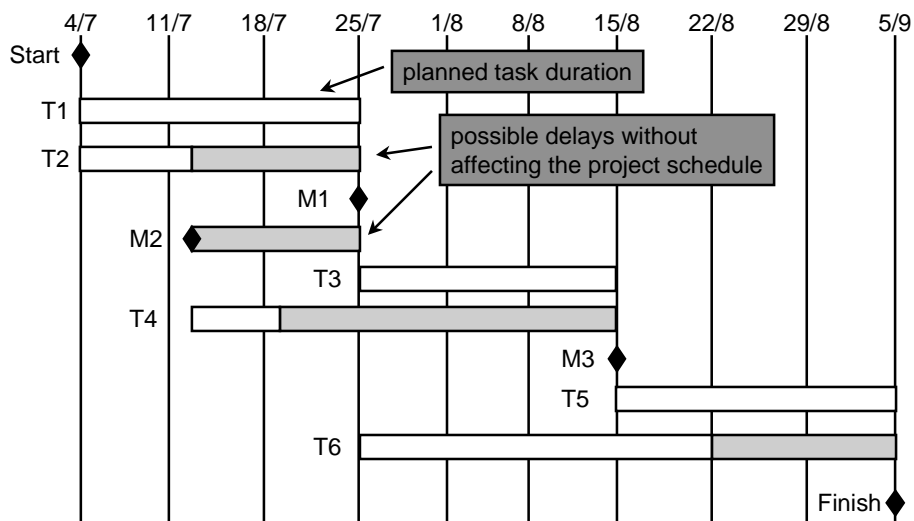


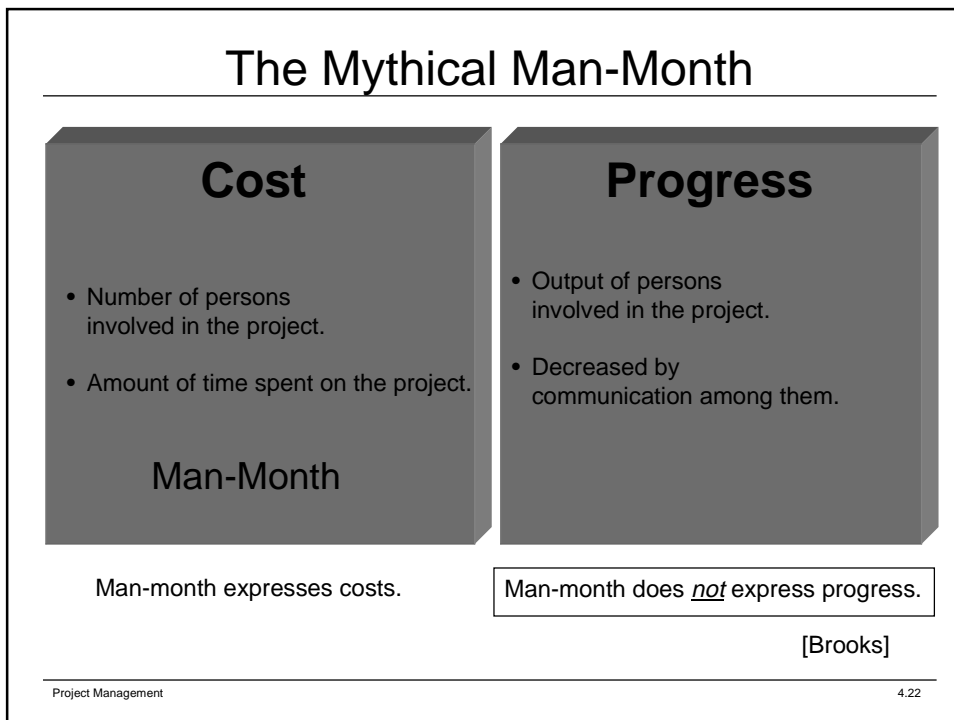
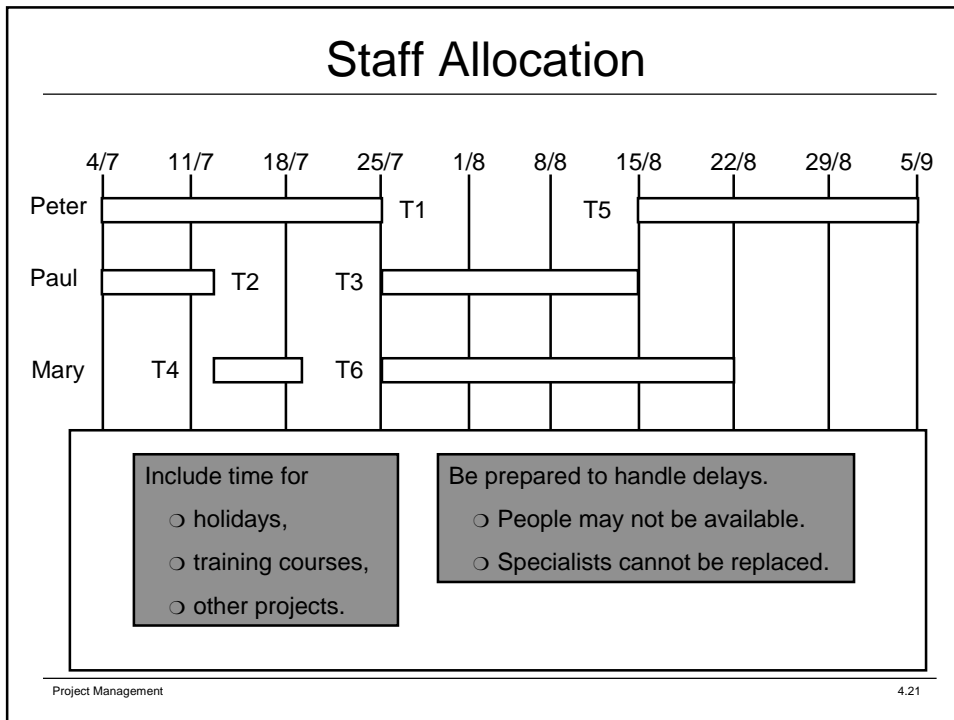
PERT Chart

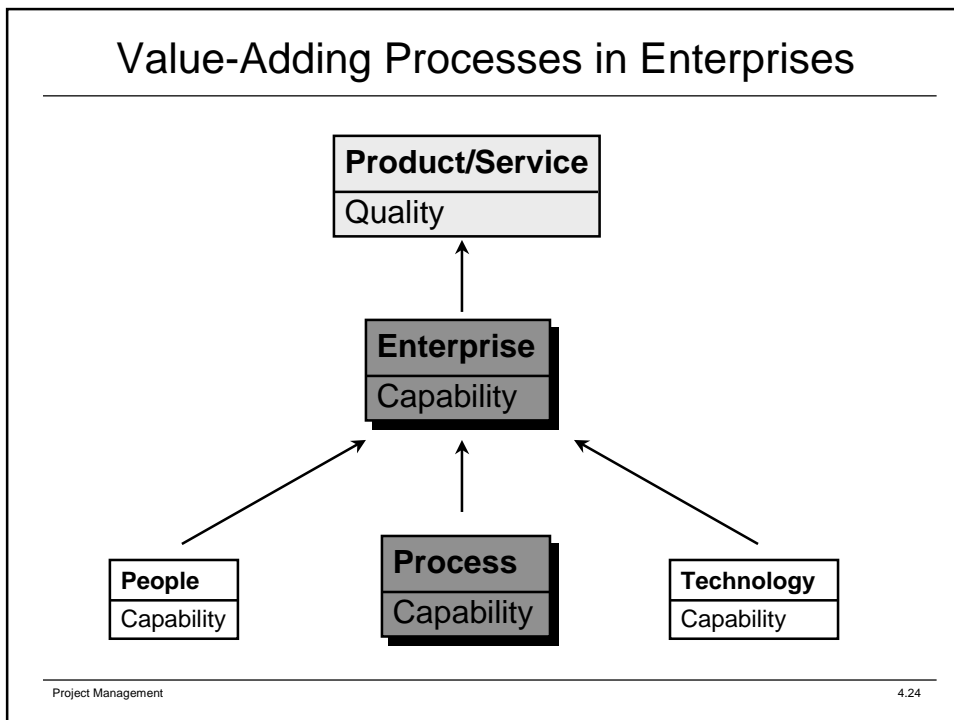
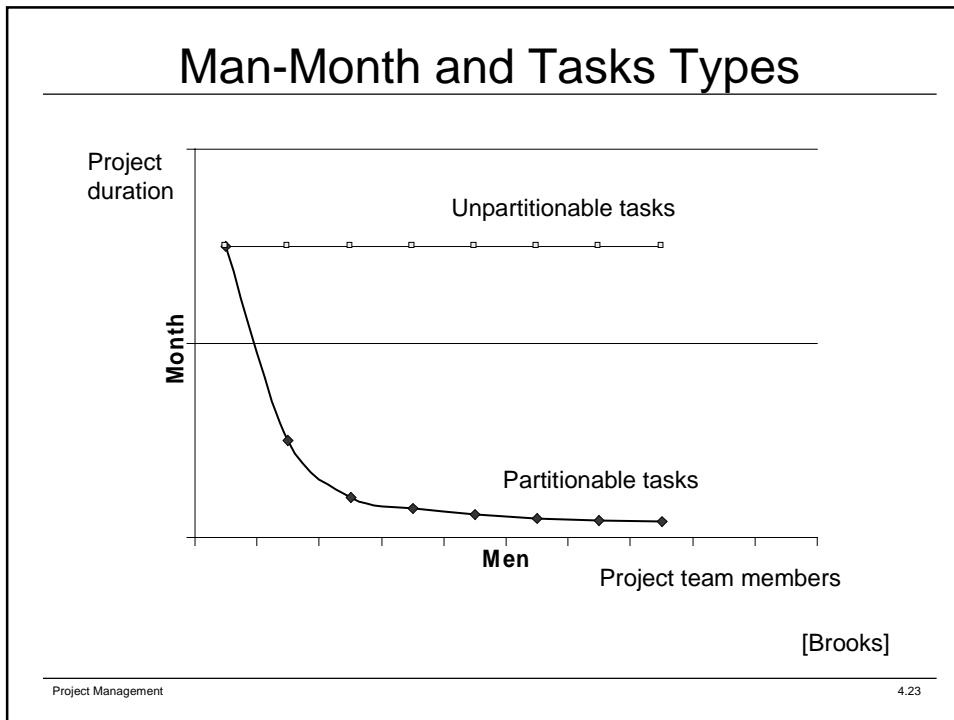
- sophisticated variant of activity networks
- distinguishes pessimistic/likely/optimistic duration/dates
- leading to many potential critical paths
- critical path analysis only with tool-support



Activity Bar Chart / Gantt chart







Process Improvement

Principles:

- You have to do it before you can manage it.
- Understand what's happening on the project (where the products are!) before defining organization-wide processes.
- Use the best of what you've learned from your projects to create organization-wide processes.
- You can't measure it until you know what "it" is.
- Managing with measurement is only meaningful when you're measuring the right things.
- A culture of continuous improvement requires a foundation of sound management practice, defined processes, and measurable goals.

Goals:

- **Predictability** of project cost and schedule.
- **Control** of performance and application of corrective actions.
- **Effectiveness**, i.e., decreasing cost, shorter development time, increasing quality and productivity.

Process Design: Organizational Context

Analyze & improve processes

- Baseline: describe current processes.
- Starting point: What are systems engineers responsible for?
- Which changes are improvements?

Understand business, product, and organizational context of process

- What life-cycle will be used as a framework for this process?
- How is the organization structured to support projects?
- How are support functions handled (e.g., by the project or the organization)?
- What are management and practitioner roles used in the organization?
- How critical are these processes to organizational success?

