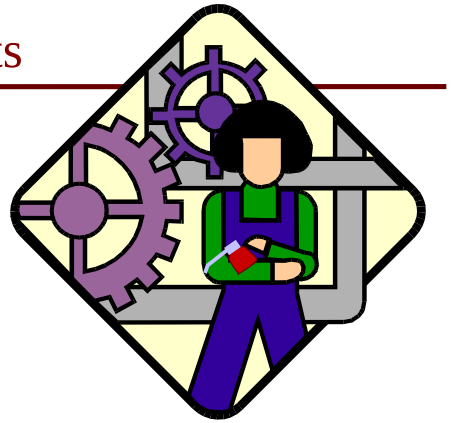
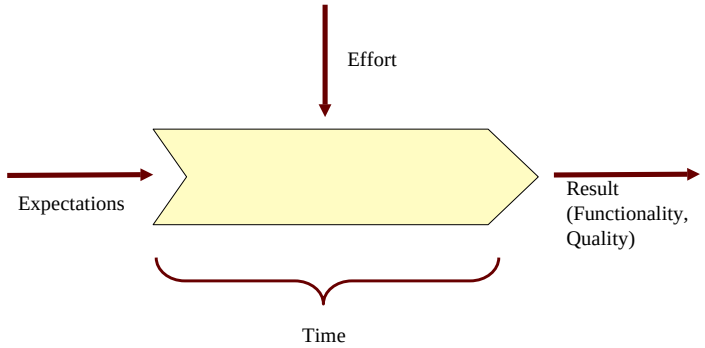
Running Projects

Controlling the Development Engine



The Project

Achieving a defined goal within limited time



The Project Management Algorithm

- Make sure we know what to do
- Figure out if it can be done, investigate alternatives
- Figure out what it will cost
- Make sure its worthwhile
- Identify required activities
- Get everything necessary to do it
- Plan necessary activities in a way that makes success as probable as possible
- Put the assets to work
- While not finished:
  - Monitor disturbances and changed expectations
  - Adjust plans to handle this
  - Acquire extra resources when needed
  - Assess that it is still worthwhile
  - Keep the sponsor informed of the state
- Deliver
- While customer not happy:
  - Fix problems
- Free all resources

"Get Everything necessary"

- People with appropriate competence
- Training
- Working premises
- Hardware and software tools
- Support (infra structure, tools, expertise)
- Travel budget
- Motivation and incentive budget

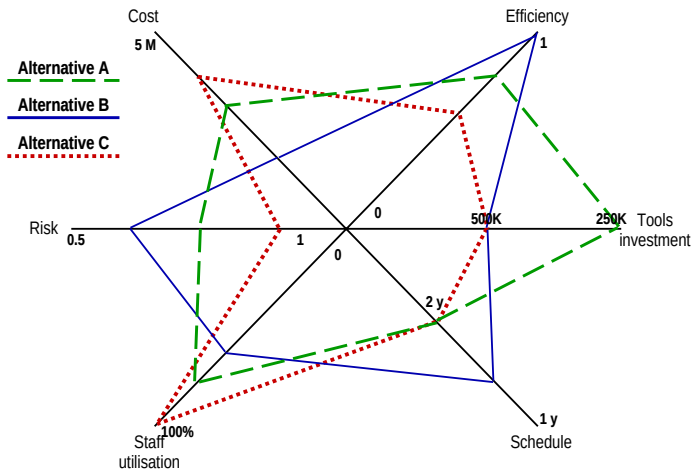
"Can it be done?"

- Compare different development alternatives
- Evaluate their risks
- Select best alternative
- Tools
  - Polar graph
  - Decision tree
  - Forms
  - ...

Analysis Example

Alternative	Cost (SEK)	Relative efficiency	Tools investment	Schedule (years)	Staff utilization (%)	Risk
A	7.500.000	0.8	250.000	2	85	0.75
B	8.500.000	1	500.000	1.3	70	0.6
C	7.000.000	0.6	500.000	2	100	0.9

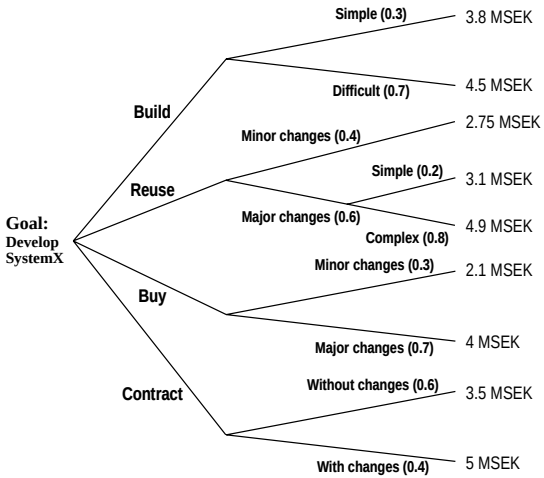
Analysis Example (Polar Graph)



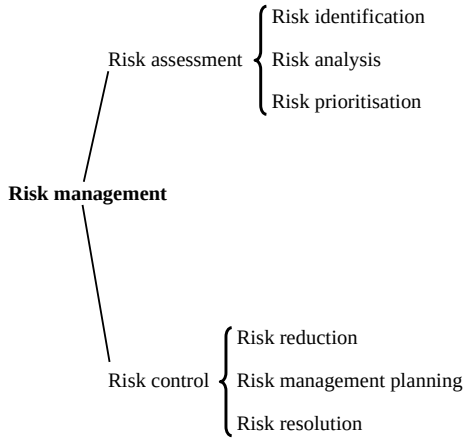
Analysis Example (Forms)

Objectives	**Develop a software components catalogue**
Constraints	Within one year Must support all existing component types Must cost less than 1 MSEK
Alternatives	Buy existing information retrieval (IR) software Buy a database and develop the catalogue using the query language Develop a special-purpose catalogue
Risks	May be impossible within the given constraints Catalogue functionality may be inappropriate
Risk resolution strategy	Develop a prototype to clarify requirements Commission a consultants report on existing IR systems Relax the time constraints
Results	IR systems are too inflexible Identified requirements cannot be met The prototype using a DBMS may be enhanced to to a complete system Special-purpose catalogue development is not cost effective
Plans	Develop the catalogue using the existing DBMS by enhancing the prototype and building a GUI
Commitment	Fund further 12 months of development

Analysis Example (Decision Tree)



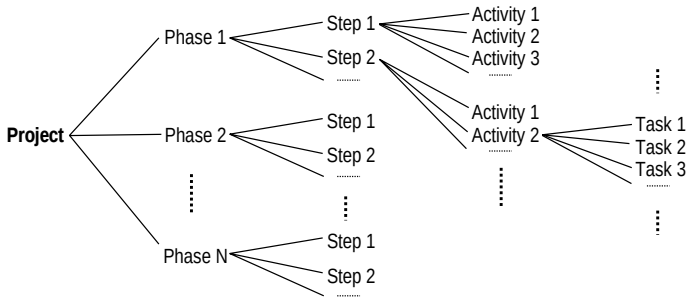
Risk Management



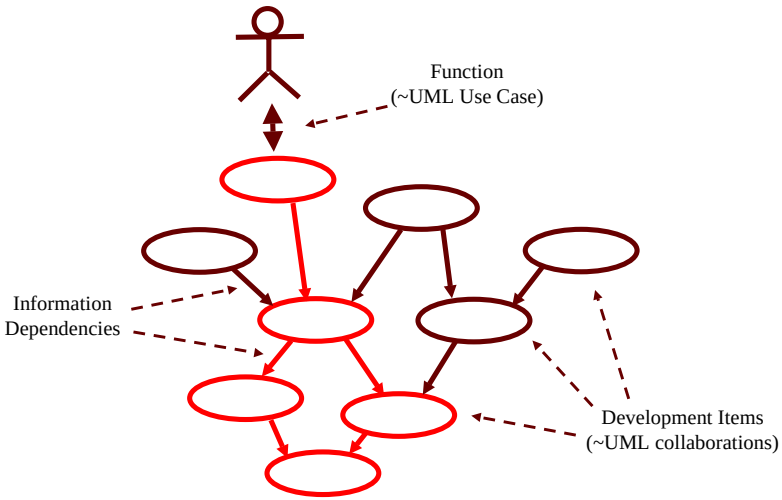
Top Ten Project Risks

- Staff deficiencies
- Unrealistic schedules and budgets
- Developing the wrong functions
- Developing the wrong interface
- Over-engineering
- Changing requirements
- Externally developed items
- Externally performed tasks
- Performance problems
- Assumptions on technology

Work Breakdown Structure



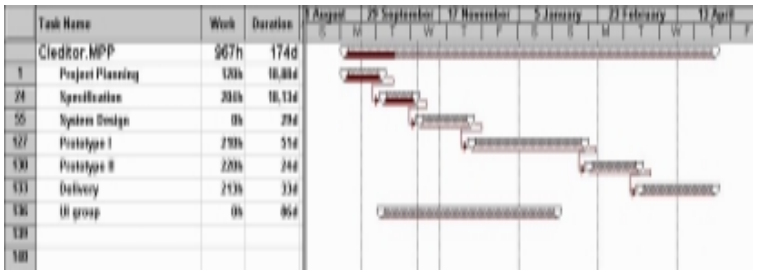
Technical Basis for Planning: The Topology of System Abilities

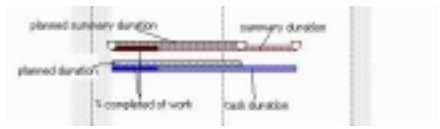


Schedule Activities

- Almost all activities depend on the completion of some other activities
- Many activities can be performed in parallel
- Track usage of resources
- ➨ Organisation necessary to balance work-load, costs, and duration

A Gantt Chart (Project Time Line)





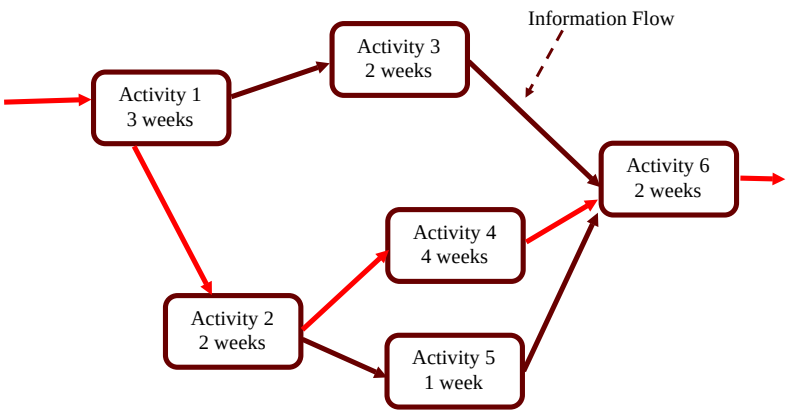
PERT Charts

Program **E**valuation and **R**eview **T**echnique

- Graph
  - Nodes = activities/tasks and estimated duration
  - Edges = dependencies
- Compute
  - Slack time = available time - estimated duration
  - Critical path

A path is critical when it contains an activity that, if delayed, will cause a delay of the whole project.

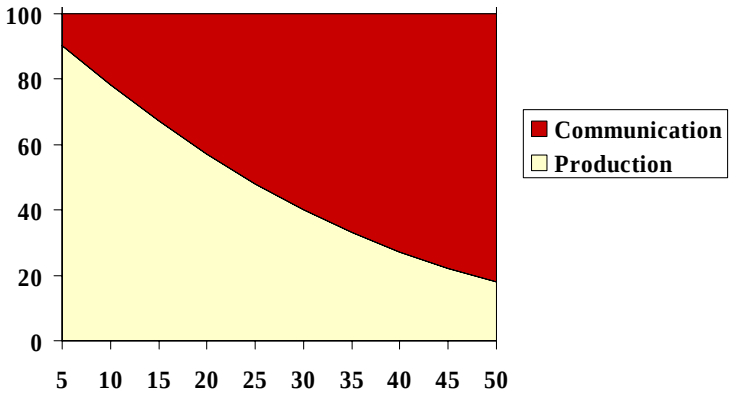
PERT Chart (Logical Plan)



Calender Time vs Man Time

Nobody can produce a baby in 1 month
by impregnating 9 women.

Productivity vs # of people



Cost Estimation

- Approach
  - Decompose problem
  - Check for experiences/ data on subproblems
  - Make qualified estimations
  - (Make at least two independent estimates)
- Problems:
  - What are good measures?
  - Do the estimates effect the result?
  - Does the type of software effect the result?
  - Does the project environment effect the result?
  - ...

Use empirical and historical data
Algorithmic cost modelling
COCOMO (based on LOC)
FP (based on *function points*)

COCOMO

- **C**onstructive **C**ost **M**odeling [Boehm 81]
- Based on publicly available historical data of 63 TRW projects
- Basic assumptions:
  - Requirements change only slightly during the project
  - There is good project management
  - The historical data is representative
  - Assigning more resources to the project does NOT result in linear decreasing development time
- Basic model:
  - $\texttt{Effort} = a \cdot (\texttt{KDSI})^b$

KDSI = Kilo Delivered Source Instructions (≈ LOC - comments)
The a and b factors vary depending on the type of project
Effort is measured in PM (Person Months = 152h of work)

COCOMO Project Types

- OM: Organic Mode projects
  - Small teams which are familiar with the type of application
  - Development in a familiar environment
- EM: Embedded Mode projects
  - Large and inexperienced teams
  - Many constraints
- SDM: Semi Detached Mode projects
  - Between OM- and EM projects

COCOMO Basic Model

- PM: Person Months
  - = $2.4\ (\texttt{KDSI})^{1.05}$ for OM projects
  - = $3\ (\texttt{KDSI})^{1.12}$ for SDM projects
  - = $3.6\ (\texttt{KDSI})^{1.20}$ for EM projects
- TDEV: Time for DEVelopment
  - = $2.5\ (\texttt{PM})^{0.38}$ for OM projects
  - = $2.5\ (\texttt{PM})^{0.35}$ for SDM projects
  - = $2.5\ (\texttt{PM})^{0.32}$ for EM projects
- N: Number of personnel
  - = PM / TDEV

This is Too Simplistic!?

- There are many *cost drivers* that effect effort
  - Programming language
  - Development methods
  - Tools and environments
  - Experience and capabilities of the development team
  - Available time
  - Requirements volatility
  - ...

COCOMO Intermediate Model

- Takes into account 15 cost drivers, which are ranked on a scale from *very low* to *extra high*
  - Product attributes (e.g. required reliability)
  - Computer system attributes (e.g. time/space constraints)
  - Personnel attributes (e.g. language experience)
  - Project attributes (e.g. tools usage)
- PM: Person Months
  - $= 3.2\ (KDSI)^{1.05} \times \Pi c_i$ for OM projects
  - $= 3\ (KDSI)^{1.12} \times \Pi c_i$ for SDM projects
  - $= 2.8\ (KDSI)^{1.20} \times \Pi c_i$ for EM projects
- $\Pi c_i \in [0.09..9.42]$
- TDEV and N as before

Intermediate COCOMO Summary

- Works quite well in practice
- TRW data is publicly available
- Needs KLOC as input
- Problems:
  - Estimating KLOC in early project stages
  - Comparison of projects using different LOC counts
  - Outdated metrics base (70s)
- Solutions:
  - Cross-check using an other estimation technique
  - Standardised LOC counts
  - Continuos model calibration

COCOMO II

- Recent new version of COCOMO
- Three stage estimation
  - Stage 1: Application Composition
    - Estimation base: Object points
    - Single standard project type
    - No cost drivers
  - Stage 2: Early Design
    - Estimation base: Function points
    - Six project type factors
    - Few cost drivers (6)
  - Stage 3: Postarchitecture
    - Estimation base: Function points or KLOC
    - Six project type factors
    - Cost drivers (16) similar to original COCOMO intermediate model

Function (Feature) Points

- Estimate functionality captured in requirements

# User inputs	x	(3,4,6)	=	
# User outputs	x	(4,5,7)	=	
# User inquiries	x	(3,4,6)	=	
# Files	x	(7,10,15)	=	
# External interfaces	x	(5,7,10)	=	
(# Algorithms	x	(3,4,6)	=	) ← Feature points only

Count-total →				

$$FP = \text{Count-total} \times [0.65 + 0.01 \times \Sigma F_i]$$

Adjustment factors ($F_i \in \{0,...,5\}$; i = 1..14)

Cost Estimation Results

"Today, a software cost estimation model is doing well if it can estimate development costs within 20% of actual costs, 70% of the time, and on its own turf (that is, within the class of projects to which it has been calibrated) ....This is not as precise as we might like, but it is accurate enough to provide a good deal of help in software engineering economic analysis and decision making."

[Boehm 81]

The Project Plan

1 Introduction
- 1.1 Project overview
- 1.2 Project deliverables
- 1.3 Evolution of the SPMP
- 1.4 Reference Materials
- 1.5 Definitions and acronyms

2 Project Organisation
- 2.1 Process model
- 2.2 Organisational structure
- 2.3 Organisational boundaries and interfaces
- 2.4 Project responsibilities

3 Managerial Process
- 3.1 Management objectives and priorities
- 3.2 Assumptions, dependencies and constraints
- 3.3. Risk management
- 3.4 Monitoring and controlling mechanisms
- 3.5 Staffing plan

4 Technical Process
- 4.1 Methods, tools and techniques
- 4.2 Software documentation
- 4.3 Project support functions

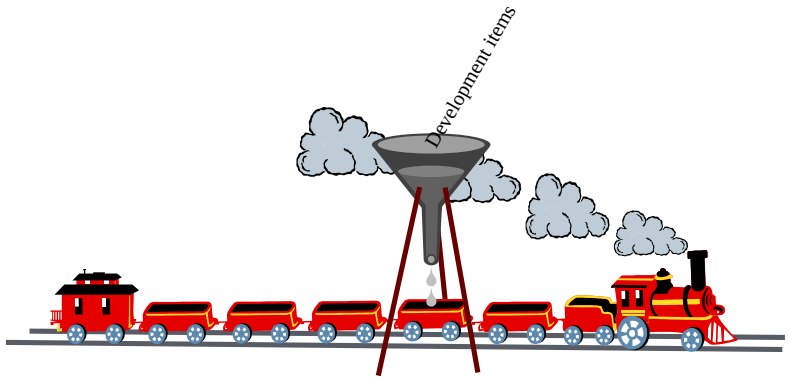
5 Work Packages, Schedule, and Budget
- 5.1 Work packages
- 5.2 Dependencies
- 5.3 Resource requirements
- 5.4 Budget and resource allocation
- 5.5 Schedule

According to ESA PSS-05-0 (see [ESA 96])

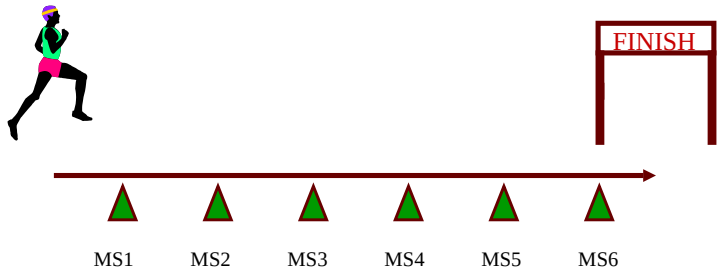
Increment Planning

- Design Items
- Architectural Impact
- Risk Reduction
- Improvements

Time Boxing



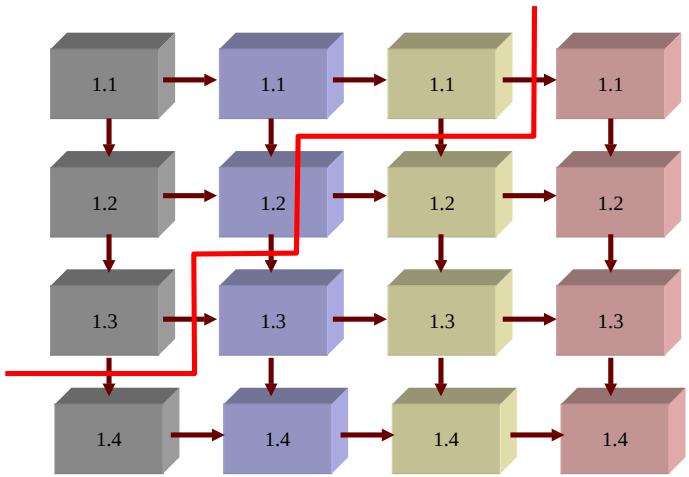
Progress Tracking



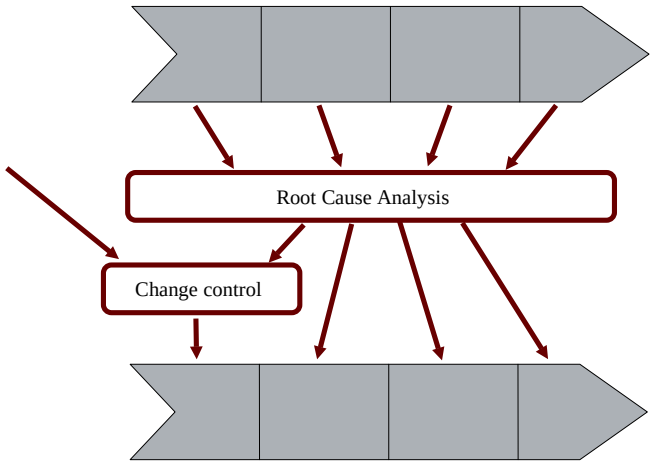
How to Measure Progress?

- Calendar time?
- Cost?
- Lines of Code?
- Gut feeling?
- Use cases?
- Objects?
- Development Items?

Progress

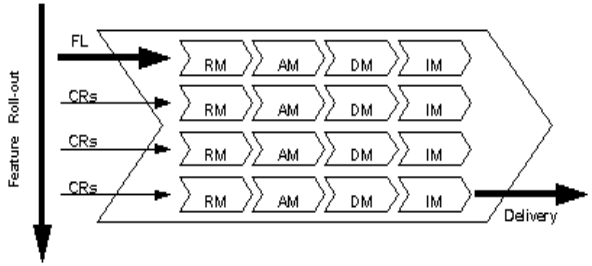


Deviation Handling



Change Request*

- Affects the system definition.
- Has to be agreed by the sponsor and the project manager.



*) There are different interpretations of this concept

Trouble Report

- Reports a fault in a previous model.
- Root cause analysis required.
- May be transformed into a change request if the fault is in the system definition.
- Fixing faults does not require an agreement.

Exemption Request

- Requests a deviation from plans.
- Practically motivated.
- Can be decided by project manager as long as project output is not affected.
- Must contain an action plan.

Discrepancy Report

- Reports that the output model is deliberately in conflict with the input model.
- Indicates that a solution in a later model has been agreed but yet not introduced in previous models.
- May be accompanied by a CR or TR.