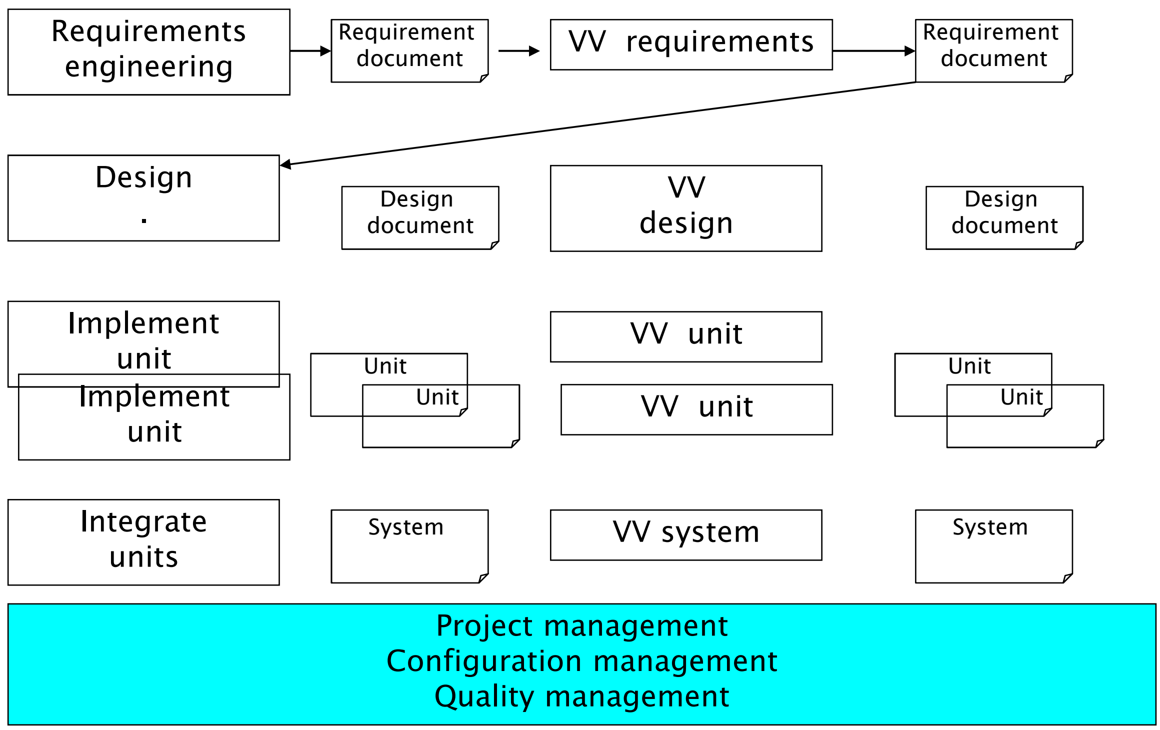
Project Management

The whole picture



Summary

- Good PM is not enough to warranty that a project succeeds
- But bad PM is enough to warranty that a project is late, over budget and does not deliver the needed functionality
- Key activities in PM are project planning, cost and effort estimation, project tracking, project organization, risk management
- Key tools are Work breakdown structure, product breakdown structure (VPM in software projects), Gantt and Pert charts, process and product measures.

Outline

Project management

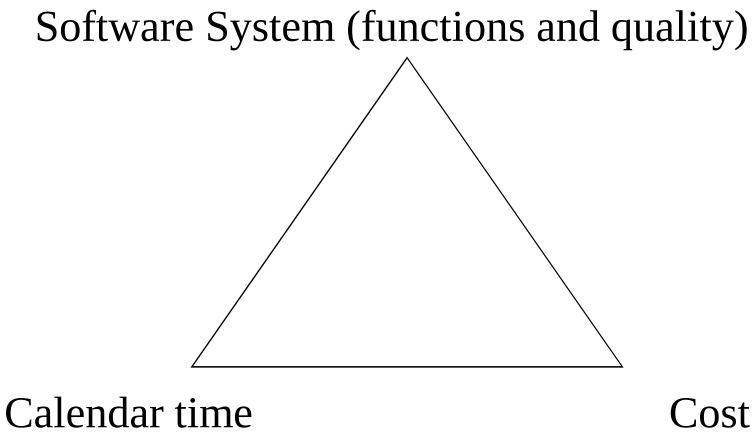
Concepts and techniques

Measures

Project planning

Risk Management

Project Management



No notion of unpredictable events here

Management activities

- planning
  - defining activities and products
  - scheduling activities and deliveries on calendar
  - deciding organizational structure
  - allocating resources
  - estimating cost / effort
- tracking
- managing risks

Concepts and techniques

Concepts and Techniques

- Concepts
  - Resource
  - Phase, Activity
  - Milestone
  - Deliverable
- Techniques
  - Pert, Gantt, WBS, PBS

Resource

- Person
- Tool

Activity, phase

- Activity
  - Time passed by resource to perform defined, coherent task
- Phase
  - Set of activities

Milestone

- Key event/condition in the project
- with effects on subsequent activities
- ex. requirement document accepted by the customer
  - if yes then ..
  - if no then ..

Deliverable

- Product (final or intermediate) in the process
  - Cfr requirements document, prototype
- internal (for producer) or external (for customer)
- contractual value or not

SOftEng http://softeng.polito.it

WBS

- Work Breakdown Structure
- Hierarchical decomposition of activities in subactivities
- no temporal relationships

Table 3.1. Phases, steps and activities of building a house.

Phase 1: Landscaping the lot	**Phase 2: Building the house**
Step 1.1: Clearing and grubbing	*Step 2.1: Prepare the site*
Activity 1.1.1: Remove trees	Activity 2.1.1: Survey the land
Activity 1.1.2: Remove stumps	Activity 2.1.2: Request permits
Step 1.2: Seeding the turf	Activity 2.1.3: Excavate for the foundation
Activity 1.2.1: Aerate the soil	Activity 2.1.4: Buy materials
Activity 1.2.2: Disperse the seeds	*Step 2.2: Building the exterior*
Activity 1.2.3: Water and weed	Activity 2.2.1: Lay the foundation
Step 1.3: Planting shrubs and trees	Activity 2.2.2: Build the outside walls
Activity 1.3.1: Obtain shrubs and trees	Activity 2.2.3: Install exterior plumbing
Activity 1.3.2: Dig holes	Activity 2.2.4: Exterior electrical work
Activity 1.3.3: Plant shrubs and trees	Activity 2.2.5: Exterior siding
Activity 1.3.4: Anchor the trees and mulch around them	Activity 2.2.6: Paint the exterior
	Activity 2.2.7: Install doors and fixtures
	Activity 2.2.8: Install roof
	Step 2.3: Finishing the interior
	Activity 2.3.1: Install the interior plumbing
	Activity 2.3.2: Install interior electrical work
	Activity 2.3.3: Install wallboard
	Activity 2.3.4: Paint the interior
	Activity 2.3.5: Install floor covering
	Activity 2.3.6: Install doors and fixtures

WBS

- Requirements planning
  - Review existing systems
  - Perform work analysis
  - Model process

Table 3.2. Milestones in building a house.

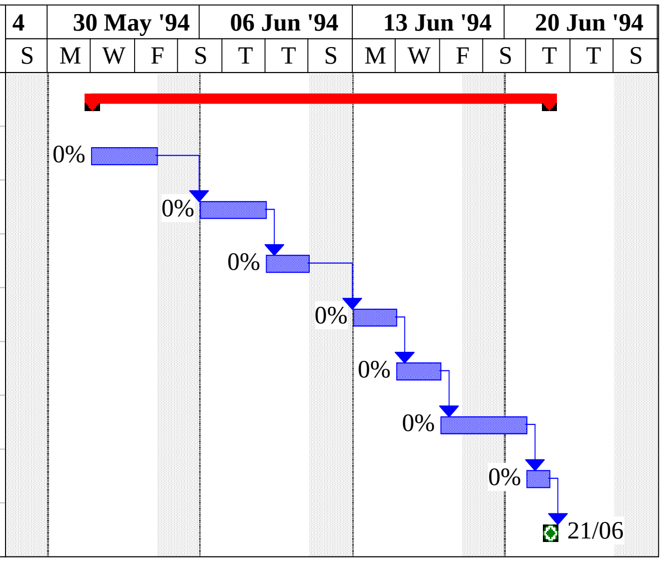
1.1.	Survey complete
1.2.	Permits issued
1.3.	Excavation complete
1.4.	Materials on hand
2.1.	Foundation laid
2.2.	Outside walls complete
2.3.	Exterior plumbing complete
2.4.	Exterior electrical work complete
2.5.	Exterior siding complete
2.6.	Exterior painting complete
2.7.	Doors and fixtures mounted
2.8.	Roof complete
3.1.	Interior plumbing complete
3.2.	Interior electrical work complete
3.3.	Wallboard in place
3.4.	Interior painting complete
3.5.	Floor covering laid
3.6.	Doors and fixtures mounted

PBS

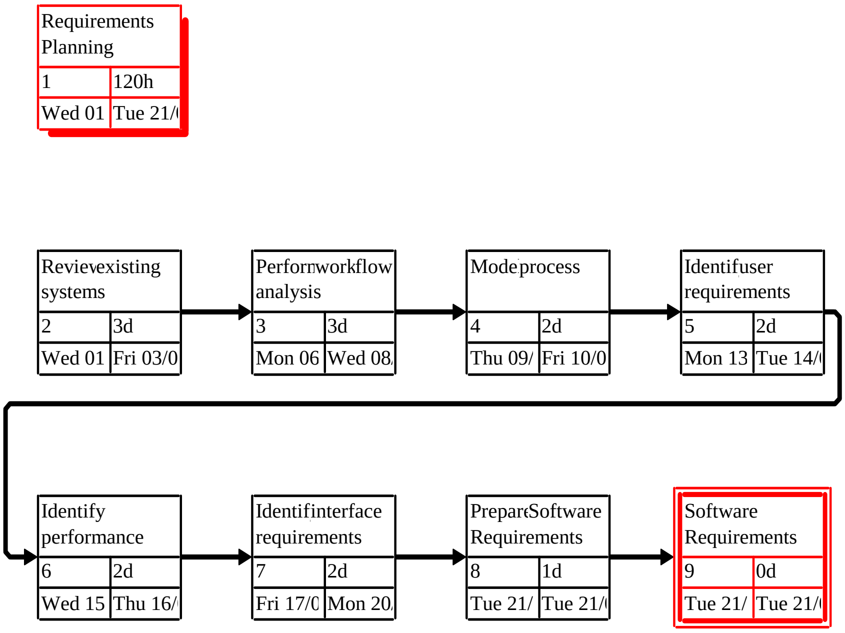
- Product Breakdown Structure
- hierarchical decomposition of product
    - Product
        - Requirement document
        - Design document
        - Module 1
            - Low level design
            - Source code
        - Module 2
            - Low level design
            - Source code
        - Testdocument

Gantt chart

ID	Task Name
1	**Requirements Planning**
2	Review existing systems
3	Perform work flow analysis
4	Model process
5	Identify user requirements
6	Identify performance requirements
7	Identify interface requirements
8	Prepare Software Requirements Specific
9	**Software Requirements Review**

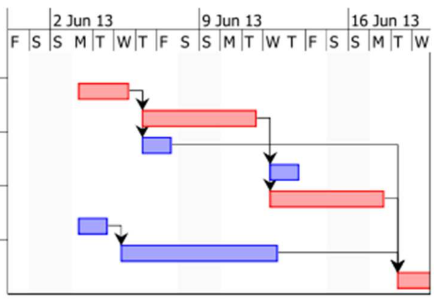


Pert

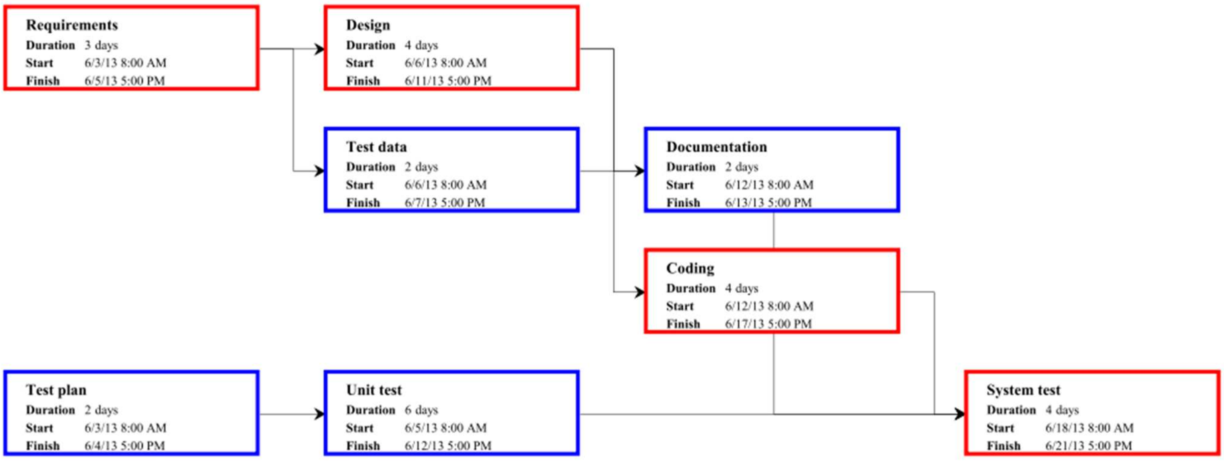


Gantt

Name	Duration	Start	Finish	Predecessors	Resource Names
Requirements	3 days	6/3...	6/5/1...		
Design	4 days	6/6...	6/11/...	1	
Test data	2 days	6/6...	6/7/1...	1	
Documentation	2 days	6/1...	6/13/...	2	
Coding	4 days	6/1...	6/17/...	2	
Test plan	2 days	6/3...	6/4/1...		
Unit test	6 days	6/5...	6/12/...	6	
System test	4 days	6/1...	6/21/...	5;3;7	



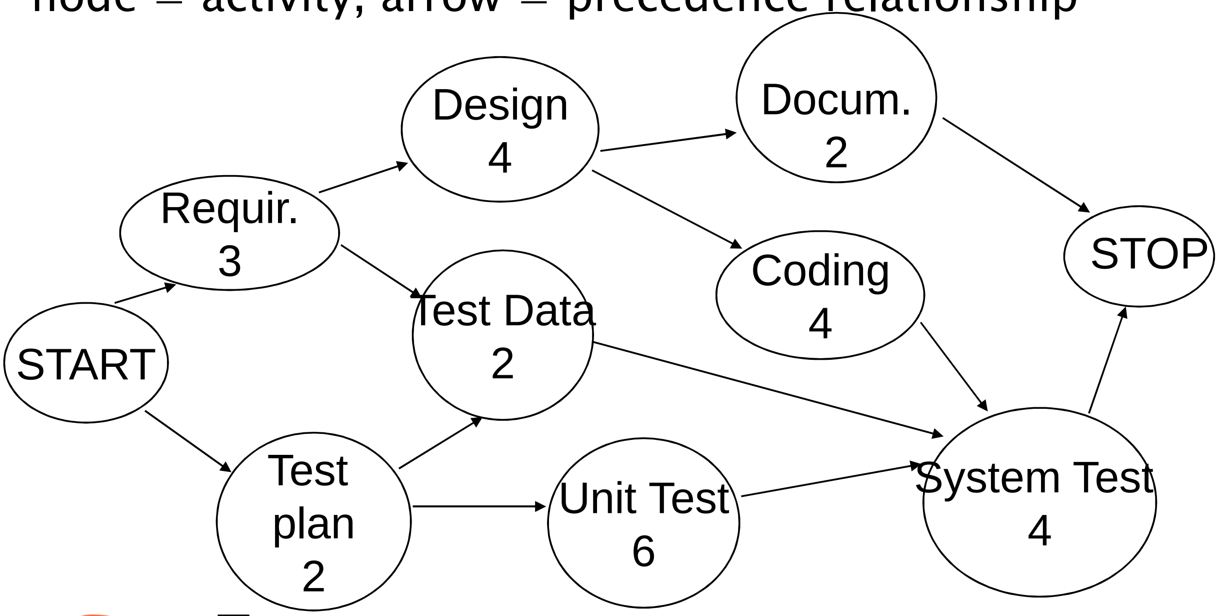
PERT



PERT

directed acyclic graph:

node = activity, arrow = precedence relationship



Critical path analysis

- What is shortest time to complete the project?
- What are the critical activities to complete the project in shortest time?
- Critical activities are the ones on the critical path(s)

Critical path

Path with longest duration

start → end

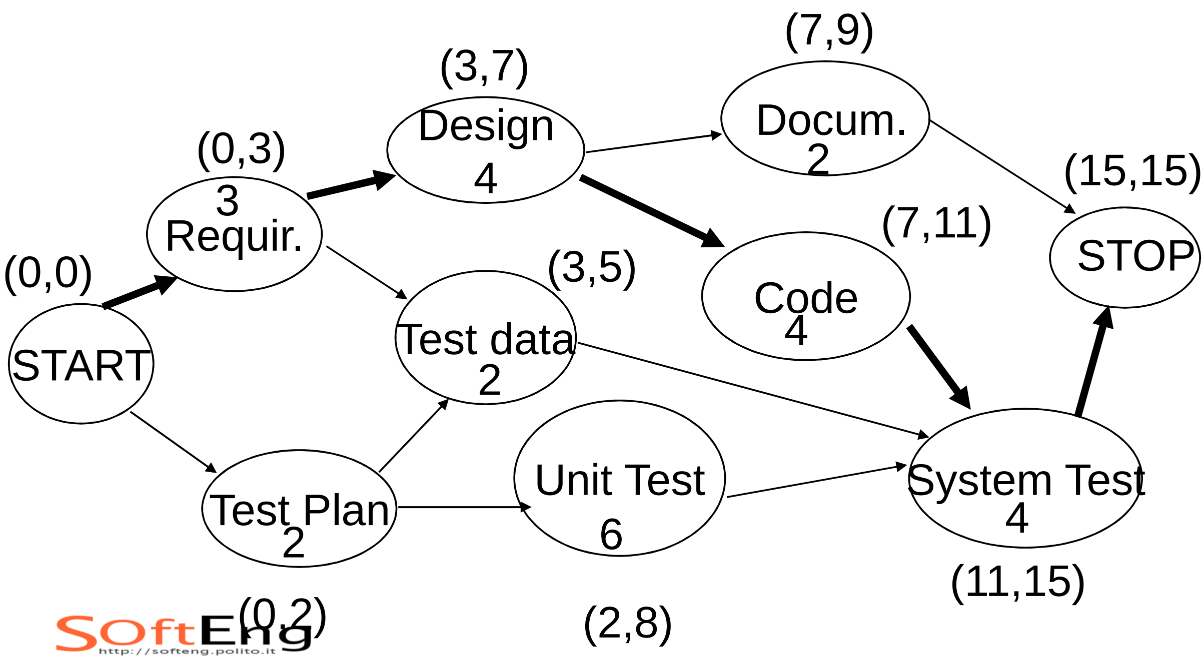
(1) START label with (0,0)

(2) For each node N whose predecessors are labeled:
SN=max {Si} Si: end time for i-th predecessor

label N with (SN, SN+duration)

(3) Repeat (2) until all nodes labeled

Example



Analysis

- Late start
  latest time an activity can be started without changing end time of project
- Slack time
  Admissible delay to complete an activity

To Compute “Slack Time”

Start from graph (S,F) from critical path analysis, for each node compute new labels (S’,F’), max start and end times

1. For STOP (S’, F’)=(S,F).
2. For each node whose successors are labeled (S’, F’) compute min S’, that becomes F’ for the node
   S’=F’-duration
   Slack Time=S’- S (or also F’- F)
3. Repeat

Managerial Implications

1. Use slack time to delay start time, or lenghten, an activity
2. If duration of activity on critical path lenghtens by X, the whole project is delayed by X
3. If only one critical path exists, reducing duration of any activity on critical path shortens duration of project.

Measures

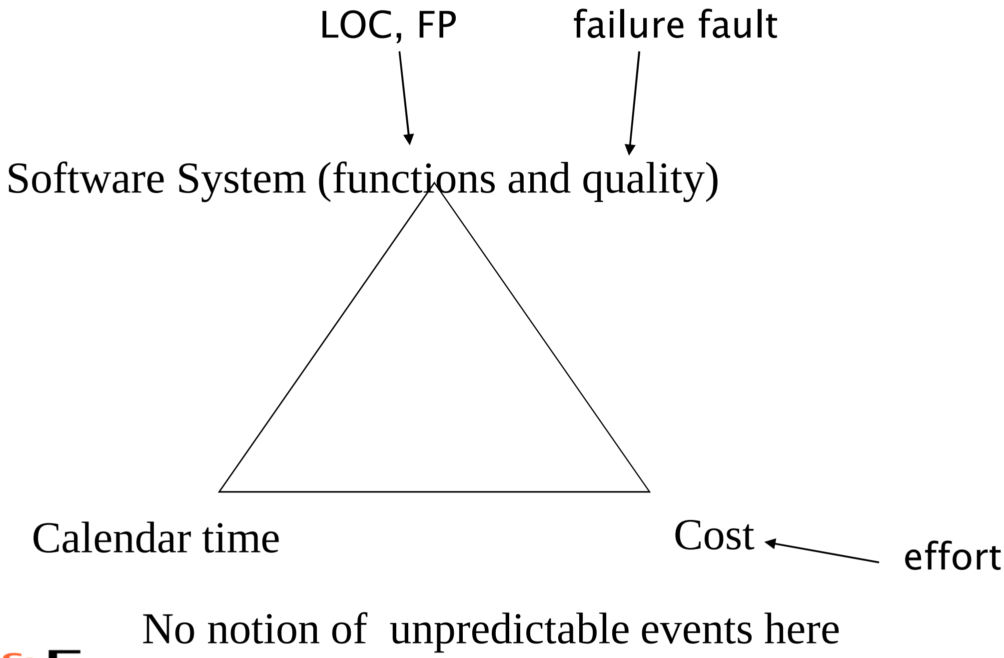
SoftEng http://softeng.polito.it

Relevant software measures

- Process measures
  - time, effort, cost
  - productivity
  - earned value
  - fault, failure, change
- Product measures
  - Functionality (FP)
  - Size
  - Price
  - Modularity
  - Other .. ilities

SoftEng http://softeng.polito.it

Measures



Calendar time, or duration

- Days, weeks, months, on calendar
- Relative, from project start
  - Month1, month2, etc
  - Typically used in planning
- Absolute
  - September 12
  - Typically used in controlling
  - Remark, transition relative -> actual is not 1 to 1 (vacations, etc)

Effort

- time taken by staff to complete a task
- Depends on calendar time and on people employed
- Measured in person hours (ieee 1045)
  - person day, person month, person year depend on national and corporation parameters

  - Converts in cost
  - Staff cost = person hours * cost per hour

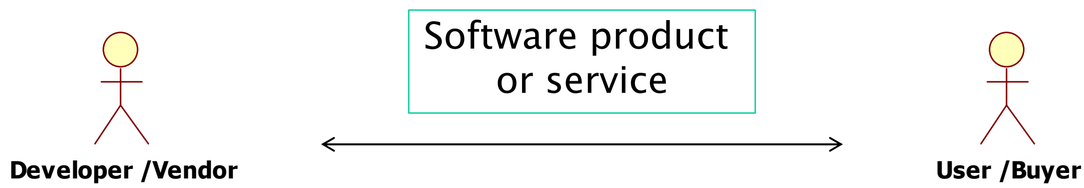
Effort

- 1 person works 6 hours → 6 ph
- 2 persons work 3 hour → 6 ph
- 6 persons work 1 hour → 6ph

Calendar time vs. effort

- Always linked
- Mathematical link. 6 ph can last
  - 6 calendar hours if 1 person works
  - 3 calendar hours if 2 persons work in parallel
  - 1 calendar hour if 6 persons work in parallel
- Practical constraint
  - Is it feasible?
    - One woman makes a baby in 9 months
    - 9 women make a baby in one month?

Costs – roles



Cost – vendor

- Personnel
  - Staff
    - Person hours, salary
    - Overhead costs (office space, heating/cooling, telephone, electricity, cleaning, ..)
  - Hardware
    - Development platform, (target platform)
  - Software
    - Licenses (OS, DB, tools ..)

Cost – user

- Total Cost of Ownership (TCO)
  - Considers the complete time window involving the product
  - At least three phases
    - Before acquisition
    - Usage
    - Dismissal

Cost – user (2)

- Before acquisition
  - Costs to define requirements and select the product
    - Market analysis, feasibility studies, requirement definition, vendor / product evaluation, contract negotiation
- Acquisition
  - Acquisition cost
    - one time fee, yearly fee, usage fee
  - Acquisition cost (= price) ⇔ vendor cost + profit

Cost – user (3)

- After acquisition
  - Deployment costs
    - Install in all users machines
    - Training for users
    - Learning curve
  - Operation costs
    - Servers, network
  - Maintenance costs
    - Collection of anomalies, effect of anomalies
    - Corrective, evolutive, enhancement maintenance

Cost – user (4)

- Dismissal
  - Uninstall product
  - Back up data, data conversion ..

TCO

- The longer the time frame, the less important the acquisition cost
  - Ex, commercial airplane
  - Time frame: 20 years (50.000 hours)
  - Cost of airplane = 1/6 of TCO
    - Key cost factors are fuel, crew, maintenance

Costing and pricing

- Estimates are made to discover the cost, to the developer, of producing a software system
- There is not a simple relationship between the development cost and the price charged to the customer
- Broader organisational, economic, political and business considerations influence the price charged

Software pricing factors

Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Size

- Of source code
  - LOC (Lines of Code)
- Of documents
  - Number of pages
  - Number of words, characters, figures, tables

Size

- Of entire project
  - Function points (see later)
  - LOC
    - In this case LOCs virtually include all documents (non code) produced in the application
    - Ex. project produces 10 documents (1000 pages) and 1000 LOCs. By convention project size is 1000 LOCs

LOC

- What to count
  - w/wout comments
  - w/wout declarations
  - w/wout blank lines
- What to include or exclude
  - Libraries, calls to services etc
  - Reused components
- Comparison for different languages not meaningful
  - C vs Java? Java vs C++? C vs ASM?

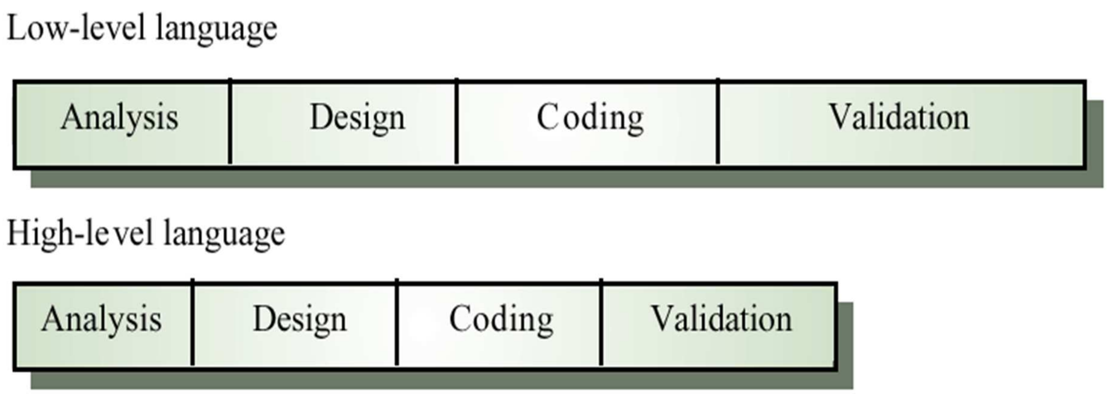
Productivity

- Output/effort
- What is output in software?
  - Size/effort = LOC / effort
    - Inherits problems of LOC
  - Functionality/effort = FP/effort
  - Object Points / effort

LOC/effort

- The lower level the language, the more productive the programmer
  - The same functionality takes more code to implement in a lower-level language than in a high-level language
- The more verbose the programmer, the higher the productivity
  - Measures of productivity based on lines of code suggest that programmers who write verbose code are more productive than programmers who write compact code

High and low level languages



Productivity paradox

	analysis	design	coding	testing	doc
Low level	3 [person weeks]	5	8	10	2
High level	3	5	4	6	2
	size	effort	productivity		
Low level	5000 [Loc]	28 [person weeks]	714 [Loc/ month]		
High level	1500	20	300		

Productivity figures

- Real-time embedded systems, 40-160 LOC/P-month
- Systems programs , 150-400 LOC/P-month
- Commercial applications, 200-800 LOC/P-month

- Source: Sommerville

Productivity figures

- Manufacturing — 0.34 FP/person hour
- Retail — 0.25
- Public administration — 0.23
- Banking — 0.12
- Insurance — 0.12

Source: Maxwell, 1999

Factors affecting productivity

Factor	Description
Application domain experience	Knowledge of the application domain is essential for effective software development. Engineers whoalready understand a domain are likely to be the most productive.
Process quality	The development process used can have a significant effect on productivity. This is covered in Chapter 31.
Project size	The larger a project, the more time required for team communications. Less time is available for development so individual productivity is reduced.
Technology support	Good support technology such as CASE tools, supportive configuration management systems, etc. can improve productivity.
Working environment	As discussed in Chapter 28, a quiet working environment with private work areas contributes to improved productivity.

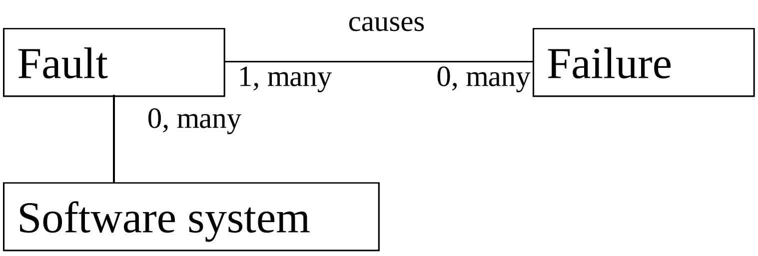
SoftEng http://softeng.polito.it

Quality and productivity

- All metrics based on size/effort are flawed because they do not take quality into account
- Productivity may generally be increased at the cost of quality
- It is not clear how productivity/quality metrics are related
- If change is constant then an approach based on counting lines of code is not meaningful

Failure vs. Fault

- Failure
  - malfunction perceived by the user
- Fault
  - defect in the system, may cause failure or not



Failure

- data to collect
  - calendar time, project time, execution time
  - effect (bad data, loss of data, ...)
  - location (product type, id)
  - gravity (human injury, economic loss, ..)
  - user profile
  - related fault(s)
- measures
  - classification, count per class
  - average time intervals

Fault

- data to collect
  - effect (related failure, if any)
  - location (product type, id)
  - type (e.g. missing req, uninitialized var, logic error, ..)
  - cause (communication, misunderstanding, clerical, ..)
  - detecting method (test, inspection, ..)
  - effort (finding and report handling)

Change

- data to collect
  - location
  - cause (related fault if corrective, adaptive, perfective)
  - effort
- measures
  - cost of failure

Fault, Failure, Change

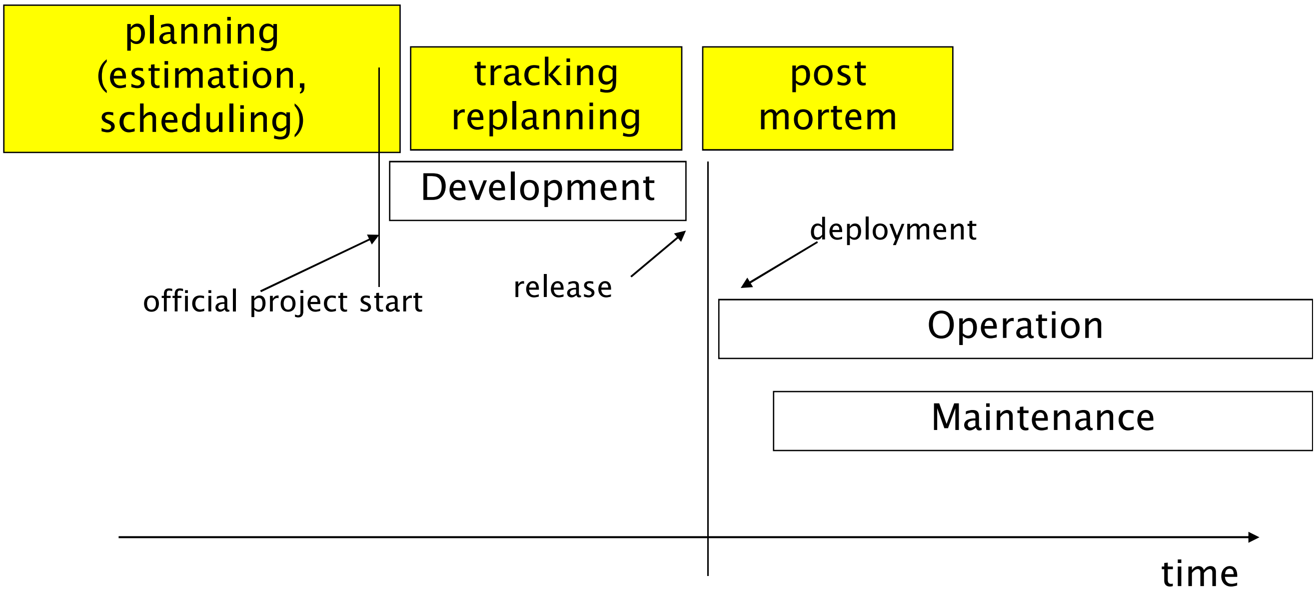
- measures
  - n open failures
  - duration/effort to close a failure
  - n failures discovered per v&v activity
  - fault/failure density
    - faults/failures per module
    - faults/failures per fp
    - faults/failures per loc
  - changes per document

Quality – Fault densities – benchmark

- Good: <1fault/1KLOC
- Bad: >10fault/1KLOC
  - Faults found in operation, 12 months after release
- Prerelease:
  - 10-30 fault/1KLOC
- Factor 10 between pre and post release

The PM process

The PM process



Planning

Planning Process

- Identify activities and/or deliverables
  - PBS, WBS
  - reference models (CMM, ISO12207)
- estimate effort and cost
- define schedule (Gantt)
- analyze schedule (Pert)

Project plan

- living document
  - will be updated during tracking
- outline
  - list of deliverables, activities
  - milestones
  - Gantt
  - Pert
  - personnel organization
  - roles and responsibilities

Estimation

SoftEng http://softeng.polito.it

Estimation of cost and effort

- Based on analogy
  - requires experience from the past to 'foresee' the future
    - Experience can be qualitative (in mind of people) or quantitative (data collected from past projects)
  - the closer a project to past projects, the better the estimation

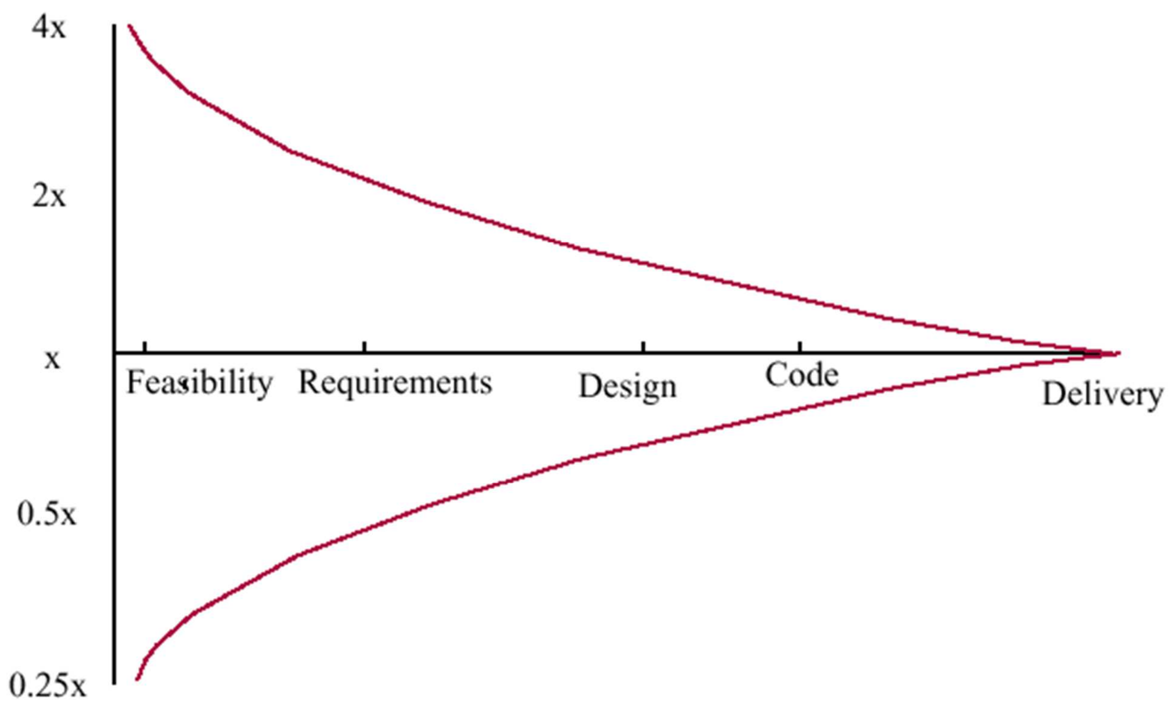
SoftEng http://softeng.polito.it

Estimation accuracy

- The cost/effort/size of a software system can only be known accurately when it is finished
- Several factors influence the final size
  - Use of COTS and components
  - Programming language
  - Distribution of system
- As the development process progresses then the estimate becomes more accurate

SoftEng http://softeng.polito.it

Estimate uncertainty



Estimation techniques

- Not suggested, but used ..
  - Parkinson's law
  - Pricing to win

Techniques – suggested

- Based on judgment
  - Decomposition
    - By activity (WBS)
    - By products (PBS)
  - Expert judgment
  - Delphi
- Based on data from the company
  - Analogy, case based
  - Regression models
- Based on data, from outside the company
  - Cocomo, Cocomo2
  - Function points
  - Object points

Parkinson's Law

- The project costs whatever resources are available
- Advantages: No overspend
- Disadvantages: System is usually unfinished

Pricing to win

- The project costs whatever the customer has to spend on it
- Advantages: You get the contract
- Disadvantages: The probability that the customer gets the system he or she wants is small. Costs do not accurately reflect the work required

By decomposition

- By activity
  - Identify activities (WBS)
  - Estimate effort per activity
  - Aggregate (linear)
- By product
  - Identify products (PBS)
  - Estimate effort per product
  - Aggregate (linear)
- Rationale: easier to estimate smaller parts

Table 3.3. Activities and time estimates.

Activity	Time estimate (in days)
Step 1: Prepare the site	
Activity 1.1: Survey the land	3
Activity 1.2: Request permits	15
Activity 1.3: Excavate for the foundation	10
Activity 1.4: Buy materials	10
Step 2: Building the exterior	
Activity 2.1: Lay the foundation	15
Activity 2.2: Build the outside walls	20
Activity 2.3: Install exterior plumbing	10
Activity 2.4: Exterior electrical work	10
Activity 2.5: Exterior siding	8
Activity 2.6: Paint the exterior	5
Activity 2.7: Install doors and fixtures	6
Activity 2.8: Install roof	9
Step 3: Finishing the interior	
Activity 3.1: Install the interior plumbing	12
Activity 3.2: Install interior electrical work	15
Activity 3.3: Install wallboard	9
Activity 3.4: Paint the interior	18
Activity 3.5: Install floor covering	11
Activity 3.6: Install doors and fixtures	7

Expert judgement

- one or more experts (chosen in function of experience) propose an estimate

Delphi

- evolution of expert judgement
- structured meetings to achieve consensus in estimate
  - each participant proposes estimate (anonymous)
  - team leader publishes synthesis
    - (a + 4m + b)/6 (beta distribution)
    - a best – b worst – m mean
  - iterate

By analogy, case based

- A set of projects
- Each project has a number of attributes (with respective values)
  - Ex size, technology, staff experience, effort, duration, etc
- Define attributes for new project
- Find 'near' project(s)
  - Distance function
- Use (adapted) effort of near project

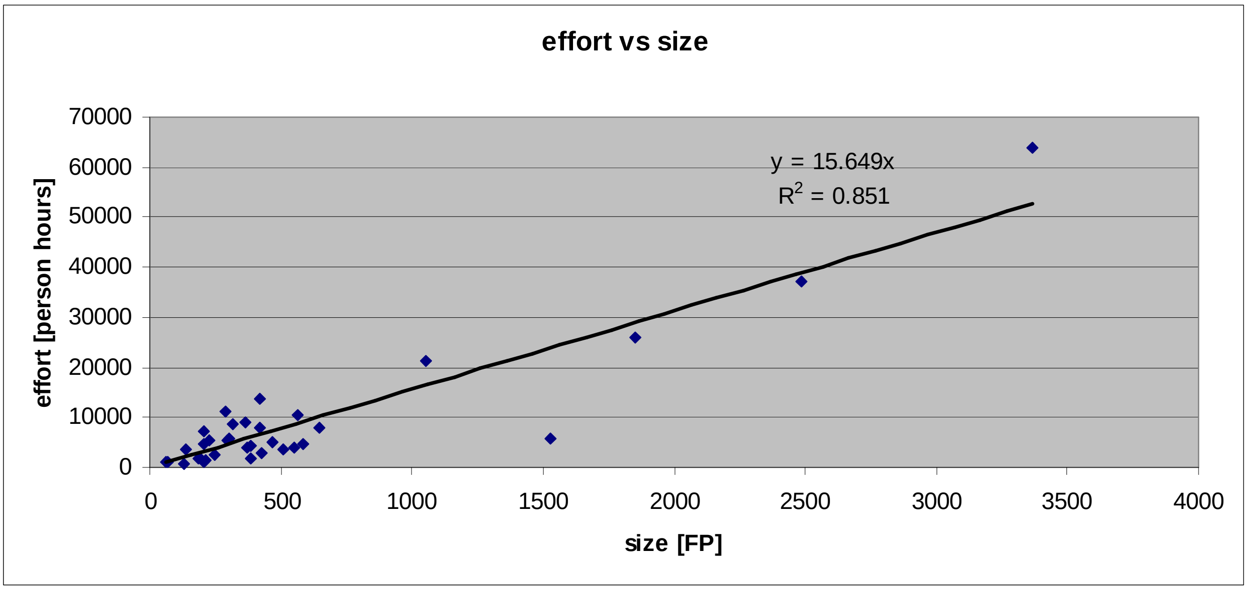
Ex.

  - See file MaxwellDataSetChap1.xls
- New project
  - We estimate
    - size = 200fp, application type =transpro, telonuse = no
  - Near projects (yellow rows) have effort
    - 7320, 1520, 963, 5578
  - We estimate effort at
    - Average of effort of yellow projects= 3845

Regression models

- If the company has a data base of past projects
  - min info required: size, effort
  - See file MaxwellDataSetChap1.xls
- apply regression (linear, or else)
- Estimate productivity
- Estimate size, compute effort

Linear regression



Ex.

- Using Maxwell data set, linear regression effort vs. size on all projects gives
  - Productivity = 1/15.649 fp/person hour
    0.063 fp per person hour
  - R2 = 0.85 (good model)
- Given new project
  - We estimate size =200fp
  - Estimated effort = 200*15.649 = 3773 ph

Function Points

- fp = A*EI + B*EO + C*EQ + D*EIF + E*ILF
  - EI = number of Input Item
  - EO = output item
  - EQ = Inquiry
  - EIF= External Interface File
  - ILF = Internal Logical File

- Coefficients A,B,C,D,E

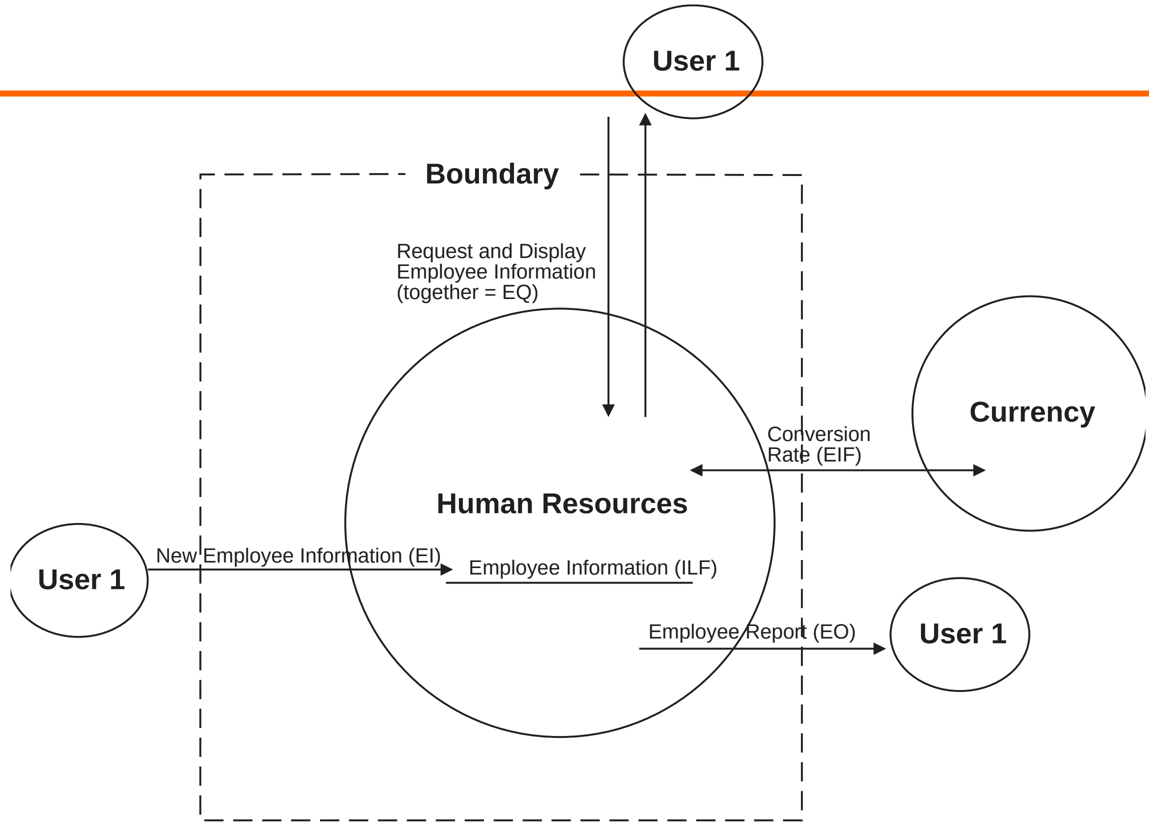
	Level of Complexity		
Component	**Simple**	**Average**	**Complex**
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

Function Points

- For any product, size in "function points" is given by

$$FP = 4 \times EI + 5 \times EO + 4 \times EQ + 10 \times ILF + 7 \times EIF$$

- A 3-step process.



Function Points (2)

- 1. Classify each component of product (EI, EO, EQ, ILF, EIF) as simple, average, or complex.
  - Assign appropriate number of function points
  - Sum gives UFP (unadjusted function points)

Function Points (3)

- 2. Compute technical complexity factor (TCF)
  - Assign value from 0 ("not present") to 5 ("strong influence throughout") to each of 14 factors such as transaction rates, portability
  - Add 14 numbers $\Rightarrow$ total degree of influence (DI)

    $$TCF = 0.65 + 0.01 \times DI$$
  - Technical complexity factor (TCF) lies between 0.65 and 1.35

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

Function Points (4)

- 3. Number of function points (FP) given by

  $$FP = UFP \times TCF$$

Function Points

- suitable for MIS
  - use of adjustment factors delicate
  - FP expert should do estimate
    - long, expensive
- conversion tables FP – LOC
  - Cobol 110
  - C 128-162
  - C++ 53-66
  - Java 53-62
- conversion tables FP – effort
  - www.ifpug.org

FP

- Advantage
  - Independent of technology
  - Independent of programmer
  - Well established and standardized
- Downside
  - Counting long and expensive
  - Transaction system oriented (no real time, no embedded systems)

FP vs. LOCS

	FP	LOCs
Depend on prog language	N	Y
Depend on programmer	N	Y
easy to compute	N, must be done by trained person	Y, tool based
Applicable to all systems	N, transaction oriented	Y

FP as unit of exchange

- Company A bids for FP
  - Buy 10000 FP, how much? (bid)
  - providers answer, x Euro per FP
- A selects provider
  - lowest cost and other factors
- End of year, redo counting
  - 10123 FP actually delivered
  - A pays

Reminder

- Measures of size
  - FP, LOC
- Both can be computed
  - Before a project start (estimated size)
  - After a project ends (actual size)
- Both can be used to
  - Characterize productivity
    - FP/effort, LOC/effort
  - Characterize application portfolio
    - FP or LOC owned and operated by a company

Function points

- IFPUG
  - FP Counting Guide
  - Exams/ certified counters
- GUFPI
- (CNIPA)

Object points

- Object points are an alternative function-related measure to function points when 4Gls or similar languages are used for development
- Object points are NOT the same as object classes

- The number of object points in a program is a weighted estimate of
  - The number of separate screens that are displayed
  - The number of reports that are produced by the system
  - The number of 3GL modules that must be developed to supplement the 4GL code

Object point estimation

- Object points are easier to estimate from a specification than function points as they are simply concerned with screens, reports and 3GL modules
- They can therefore be estimated at an early point in the development process. At this stage, it is very difficult to estimate the number of lines of code in a system

The COCOMO model

- 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 81

- Based on 63 project
  - Various types: scientific, MIS, embedded
  - Data set then enriched
- Assumes waterfall process
  - Planning and requirements analysis
  - Design
  - Implementation
  - Integration and test
- Estimate covers 3 latter phases

COCOMO 81

Project complexity	Formula	Description
Simple	$PM = 2.4\ (KDSI)^{1.05} \times M$	Well-understood applications developed by small teams.
Moderate	$PM = 3.0\ (KDSI)^{1.12} \times M$	More complex projects where team members may have limited experience of related systems.
Embedded	$PM = 3.6\ (KDSI)^{1.20} \times M$	Complex projects where the software is part of a strongly coupled complex of hardware, software, regulations and operational procedures.

Base model

- PM = effort in person months
- KDSI = K Delivered Source Instructions
- M = 1

Intermediate model

Cost Drivers	Rating: Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
Personnel Attributes						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Programmer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project Attributes						
Use of modern programming practices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

*For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system) it calls on to accomplish its task.

M, example

Cost Drivers	Situation	Rating	Effort Multiplier
Required software reliability	Serious financial consequences of software fault	High	1.15
Database size	20,000 bytes	Low	0.94
Product complexity	Communications processing	Very high	1.30
Execution time constraint	Will use 70% of available time	High	1.11
Main storage constraint	45K of 64K store (70%)	High	1.06
Virtual machine volatility	Based on commercial microprocessor hardware	Nominal	1.00
Computer turnaround time	Two hour average turnaround time	Nominal	1.00
Analyst capabilities	Good senior analysts	High	0.86
Applications experience	Three years	Nominal	1.00
Programmer capability	Good senior programmers	High	0.86
Virtual machine experience	Six months	Low	1.10
Programming language experience	Twelve months	Nominal	1.00
Use of modern programming practices	Most techniques in use over one year	High	0.91
Use of software tools	At basic minicomputer tool level	Low	1.10
Required development schedule	Nine months	Nominal	1.00

COCOMO 2 (1997) levels

- a 3 level model that allows increasingly detailed estimates to be prepared as development progresses
- Early prototyping level
  - Estimates based on object points and a simple formula is used for effort estimation
- Early design level
  - Estimates based on function points that are then translated to LOC
- Post-architecture level
  - Estimates based on lines of source code

Early prototyping level

- Supports prototyping projects and projects where there is extensive reuse
- Based on standard estimates of developer productivity in object points/month
- Takes CASE tool use into account

- Formula is
  - $PM = (NOP \times (1 - \%reuse/100)) / PROD$
  - PM is the effort in person-months, NOP is the number of object points and PROD is the productivity

Object point productivity

Developer's experience and capability	Very low	Low	Nominal	High	Very high
ICASE maturity and capability	Very low	Low	Nominal	High	Very high
PROD (NOP/month)	4	7	13	25	50

Early design level

- Estimates can be made after the requirements have been agreed
- Based on standard formula for algorithmic models
  - $PM = A \times Size^B \times M + PM_m$ where
  - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
  - $PM_m = (ASLOC \times (AT/100)) / ATPROD$

- A = 2.5 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity

Multipliers

- RCPX – product reliability and complexity
- RUSE – the reuse required
- PDIF – platform difficulty
- PREX – personnel experience
- PERS – personnel capability
- SCED – required schedule
- FCIL – the team support facilities

- PM reflects the amount of automatically generated code

Post-architecture level

- Uses same formula as early design estimates
- Estimate of size is adjusted to take into account
  - Requirements volatility. Rework required to support change
  - Extent of possible reuse. Reuse is non-linear and has associated costs so this is not a simple reduction in LOC
  - ESLOC = ASLOC × (AA + SU +0.4DM + 0.3CM +0.3IM)/100

SoftEng http://softeng.polito.it

– ESLOC is equivalent number of lines of new code. ASLOC is the number of lines of reusable code which must be modified, DM is the percentage of design modified, CM is the percentage of the code that is modified , IM is the percentage of the original integration effort required for integrating the reused software.

– SU is a factor based on the cost of software understanding, AA is a factor which reflects the initial assessment costs of deciding if software may be reused.

SoftEng http://softeng.polito.it

The exponent term

- This depends on 5 scale factors (see next slide). Their sum/100 is added to 1.01
- Example
  - Precedenteness – new project – 4
  - Development flexibility – no client involvement – Very high – 1
  - Architecture/risk resolution – No risk analysis – V. Low – 5
  - Team cohesion – new team – nominal – 3
  - Process maturity – some control – nominal – 3
- Scale factor is therefore 1.17

Exponent scale factors

Scale factor	Explanation
Precedentedness	Reflects the previous experience of the organisation with this type of project. Very low means no previous experience, Extra high means that the organisation is completely familiar with this application domain.
Development flexibility	Reflects the degree of flexibility in the development process. Very low means a prescribed process is used; Extra high means that the client only sets general goals.
Architecture/risk resolution	Reflects the extent of risk analysis carried out. Very low means little analysis, Extra high means a complete a thorough risk analysis.
Team cohesion	Reflects how well the development team know each other and work together. Very low means very difficult interactions, Extra high means an integrated and effective team with no communication problems.
Process maturity	Reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level from 5.

Multipliers

- Product attributes
  - concerned with required characteristics of the software product being developed
- Computer attributes
  - constraints imposed on the software by the hardware platform
- Personnel attributes
  - multipliers that take the experience and capabilities of the people working on the project into account.
- Project attributes
  - concerned with the particular characteristics of the software development project

Project cost drivers

Product attributes			
RELY	Required system reliability	DATA	Size of database used
CPLX	Complexity of system modules	RUSE	Required percentage of reusable components
DOCU	Extent of documentation required		
Computer attributes			
TIME	Execution time constraints	STOR	Memory constraints
PVOL	Volatility of development platform		
Personnel attributes			
ACAP	Capability of project analysts	PCAP	Programmer capability
PCON	Personnel continuity	AEXP	Analyst experience in project domain
PEXP	Programmer experience in project domain	LTEX	Language and tool experience
Project attributes			
TOOL	Use of software tools	SITE	Extent of multi-site working and quality of site communications
SCED	Development schedule compression		

Effects of cost drivers

Exponent value System size (including factors for reuse and requirements volatility) **Initial COCOMO estimate without cost drivers**	1.17 128, 000 DSI **730 person-months**
Reliability Complexity Memory constraint Tool use Schedule **Adjusted COCOMO estimate**	Very high, multiplier = 1.39 Very high, multiplier = 1.3 High, multiplier = 1.21 Low, multiplier = 1.12 Accelerated, multiplier = 1.29 **2306 person-months**
Reliability Complexity Memory constraint Tool use Schedule **Adjusted COCOMO estimate**	Very low, multiplier = 0.75 Very low, multiplier = 0.75 None, multiplier = 1 Very high, multiplier = 0.72 Normal, multiplier = 1 **295 person-months**

Sw project Data sets

- Company specific
  - When exists
  - Maxwell, Applied statistics for software managers, Prentice Hall
- Public
  - Knowledge plan (Caper Jones)
  - Software productivity research
  - ISBSG, Int. software benchmarking standards group, www.isbsg.org

Scheduling

Project duration

- As well as effort estimation, calendar time must be estimated, and staff allocated
- Scheduling can be done on Gantt/Pert
- COCOMO2 gives also an estimate of calendar time
  - Independent of staffing

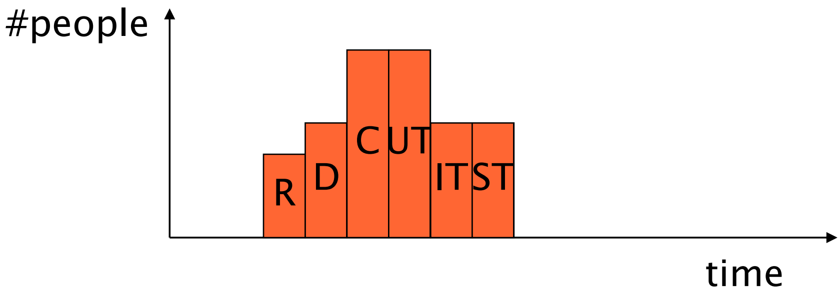
- Calendar time can be estimated using a COCOMO 2 formula
  - $TDEV = 3 \times (PM)^{(0.33+0.2*(B-1.01))}$
  - PM is the effort computation and B is the exponent computed as discussed above (B is 1 for the early prototyping model). This computation predicts the nominal schedule for the project

Staffing requirements

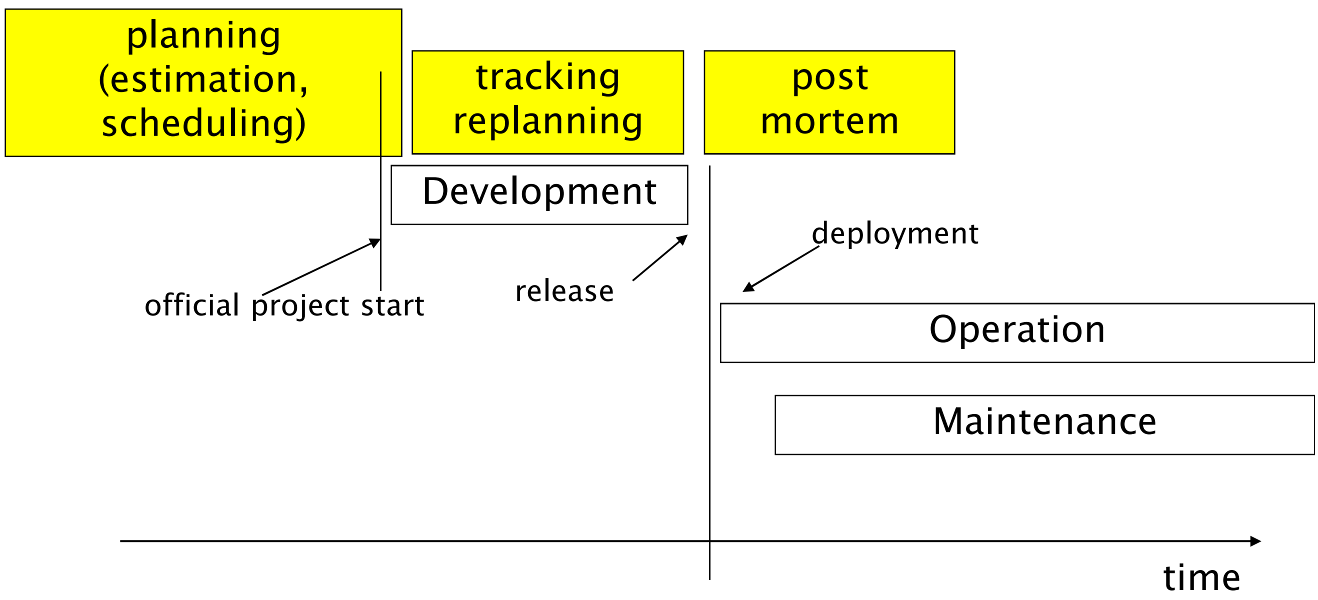
- Staff required can't be computed by dividing the development time by the required schedule
- The number of people working on a project varies depending on the phase of the project
- The more people who work on the project, the more total effort is usually required
- A very rapid build-up of people often correlates with schedule slippage

Staffing profile

- Number of people working on the project vs. time
- Typically has a bell shape
  - duration of project is constrained by staffing profile + total effort estimated



The PM process



Tracking

Tracking process

- project has started how to know status of project?
- collect project data, define actual status
- compare estimated – actual
  - Estimated Gantt is the roadmap for project
- if deviations, do corrective actions
  - change personnel, change activities, change deliverables, ...
  - re-plan, update Gantt and PERT

Project status

- Option1
  - Effort spent
- Option2
  - Effort spent + activities closed
- Option3
  - Earned value

Effort spent

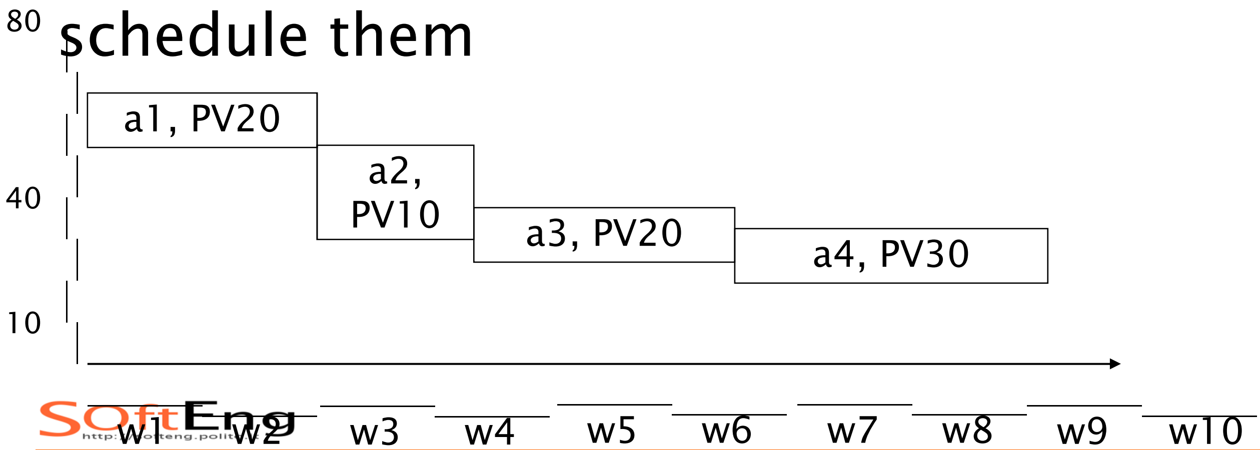
- Collect effort spent, compare with estimated
  - Ex, spent 10, estimated 100, we are done 10%
- Big flaw, confounds input measure (effort spent) with output measure (completion)
  - Typical result, spent 90, estimated 100, but the remaining 10% takes 100..

Activities closed

- How to define when activity is closed?
  - All effort planned for activity is spent
    - Same problem, confounds input with output
  - Define quality gate, level to achieve
    - Ex, requirements: inspection meeting, majority of participants judges document is goodenough
    - Ex, unit testing: coverage 95% of nodes, and all tests pass

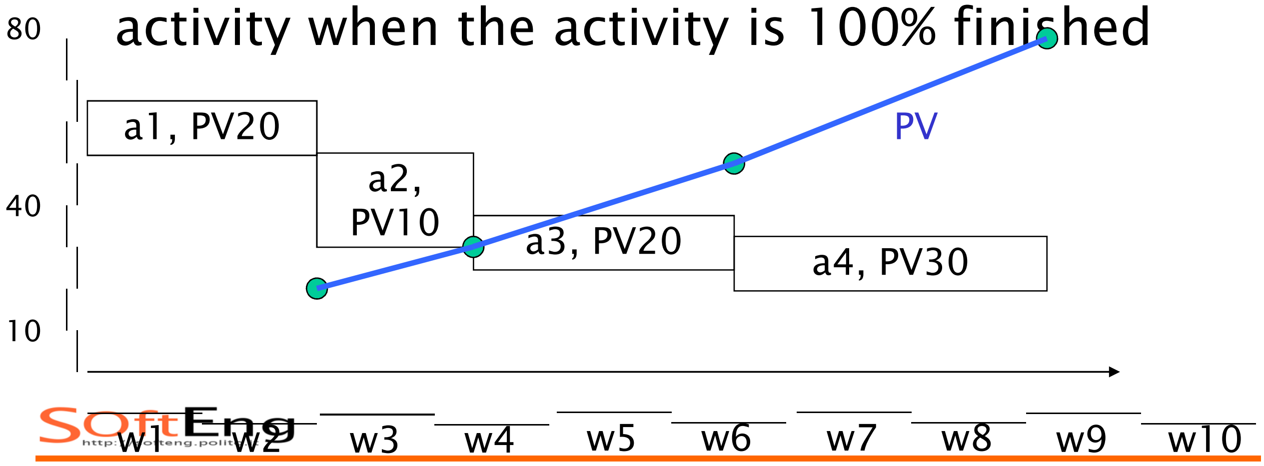
Earned value

- A technique to measure progress of a project
- Step 1: identify activities, assign a value to them (Planned Value, PV), schedule them



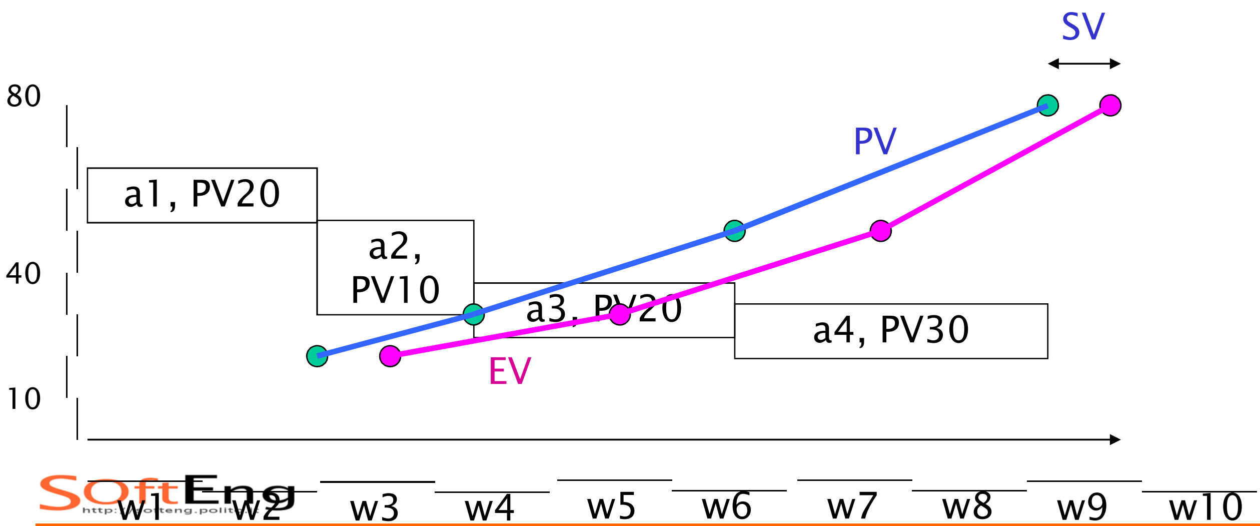
Earned value

- Step 2: define a rule to pass from PV to EV (rule1 0/100 or rule2 0/50/100)
  - With rule1, the project earns the PV of an activity when the activity is 100% finished



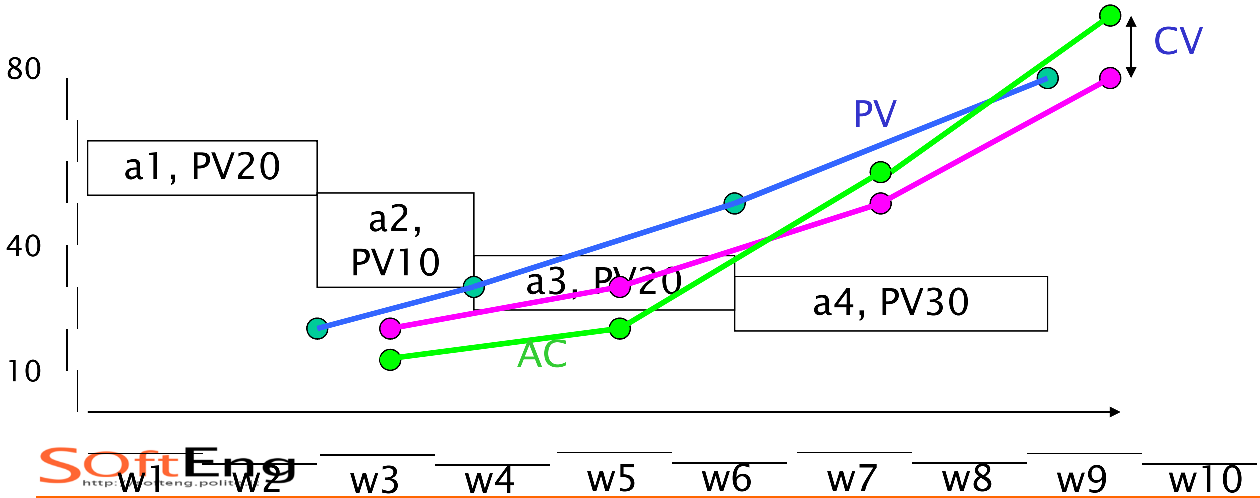
Earned value

- Step 3: start the project, measure EV and compare with PV

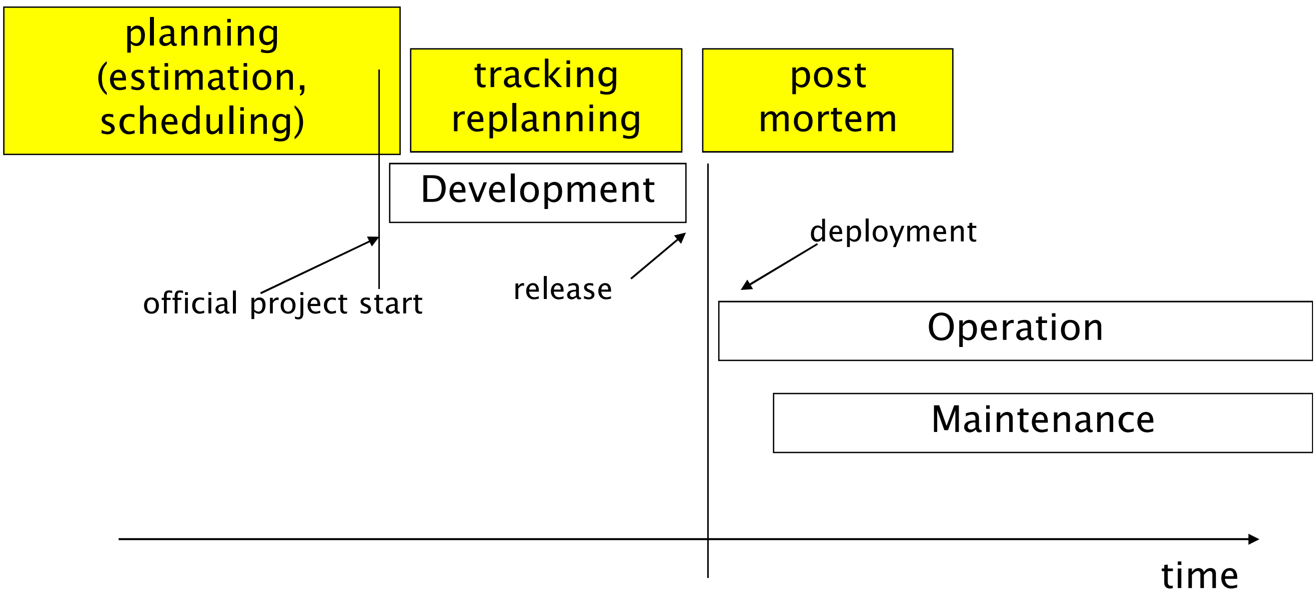


Earned value

- Step 4: compute also AC, actual cost



The PM process



Post Mortem

SoftEng http://softeng.polito.it

Post mortem

- A form of organizational learning
- Collect key information from the project
  - Effort, faults – estimated and actual
  - Achievements
  - Problems and causes
- To make it available to other projects

SoftEng http://softeng.polito.it

PMA – learn from experience

- PMA (when used appropriately) PMA ensures that team members recognise and remember what they learned during a project.
- PMA identifies improvement opportunities and provides means to initiate sustained change.
- PMA provides qualitative feedback
- Two types
  - General PMA
  - Focused PMA – understanding and improving a project`s specific activity

PMA process

- Preparation
  - Study the project history to understand what has happened
  - Review all available documents
  - Determine goal for PMA
  - Example of goal: Identify major project achievements and further improvement opportunities.

PMA process cont.

- Data collection
  - Gather relevant project experience
  - Focus on positive and negative aspects
  - Semistructured interviews – pre-prepared list of questions
  - Facilitated group discussion
  - KJ sessions
    - Write down up to four positive and negative project experience on post-it notes.
    - Put the notes on a whiteboard
    - Re-arrange notes into groups and discuss them

PMA process cont.

- Analysis
  - Feedback session
    - Have we (analyser) understood what you (project member) told us, and do we have all the relevant facts?
  - Ishikawa diagram in a collaborative process to find the causes for positive and negative experiences
    - Draw an arrow on a whiteboard – which is label with experience
    - Add arrows with causes (the diagram will look like a fishbone)

PMA – results and experience

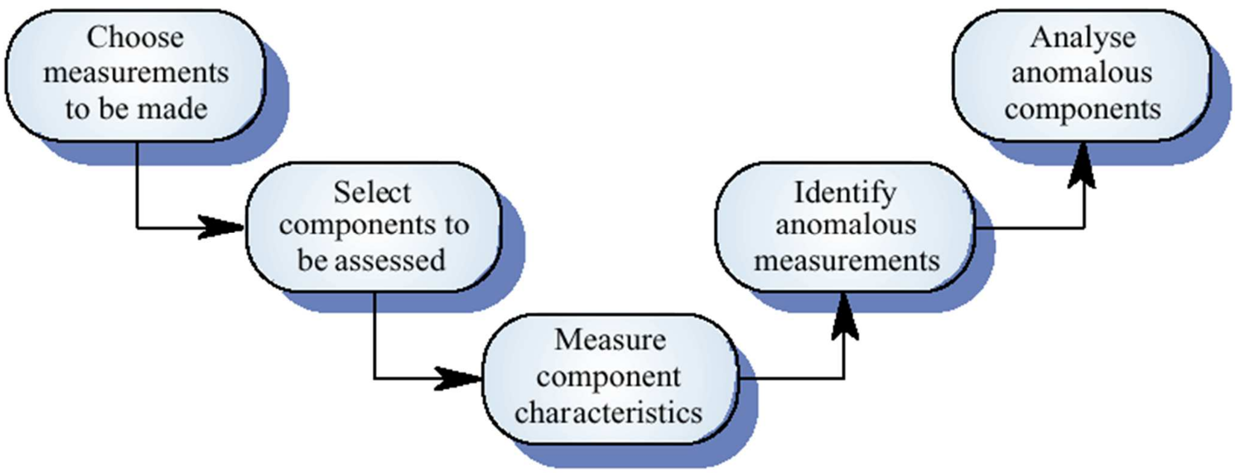
- Document the PMA results in a project experience report
  - Project description
  - Projects main problems, with description and Ishikawa diagrams
  - Project main success, with descriptions and Ishikawa diagrams
  - PMA meeting as an appendix (to let the reader see how the team discussed problems and successes)

Collecting and using measures

The measurement process

- A process should be defined and implemented to collect data, derive and analyze measures
- Data collected during this process should be maintained as an organisational resource
- Once a measurement database has been established, comparisons across projects become possible

Product measurement process



GQM

- Focus on few, important measures (top down)
- Never “collect everything, analyze later” (bottom up)
  - Too much data
  - Not meaningful

Goal – (similar to KPI)

- G1Satisfying customer
  - What is satisfaction?
    - Interviews
  - What is quality of product?
    - Defects after delivery
- G2 produce low cost product
  - What is cost
    - Cost of development

SoftEng http://softeng.polito.it

Typical indicators

- Effort (Cost)
- Size
- Defects after delivery
- Defects during development

GQM example

- Overall research question
  - Are UML Object diagrams useful?

Goal

- Object of study
  - UML Static structure diagrams
- Purpose
  - Evaluate
- Focus
  - Usefulness
- Point of view
  - Maintainer comprehending software
- Context
  - Master degree class

Data collection

- A metrics programme should be based on a set of product and process data
- Data should be collected immediately (not in retrospect) and, if possible, automatically
- Data should be controlled and validated as soon as possible

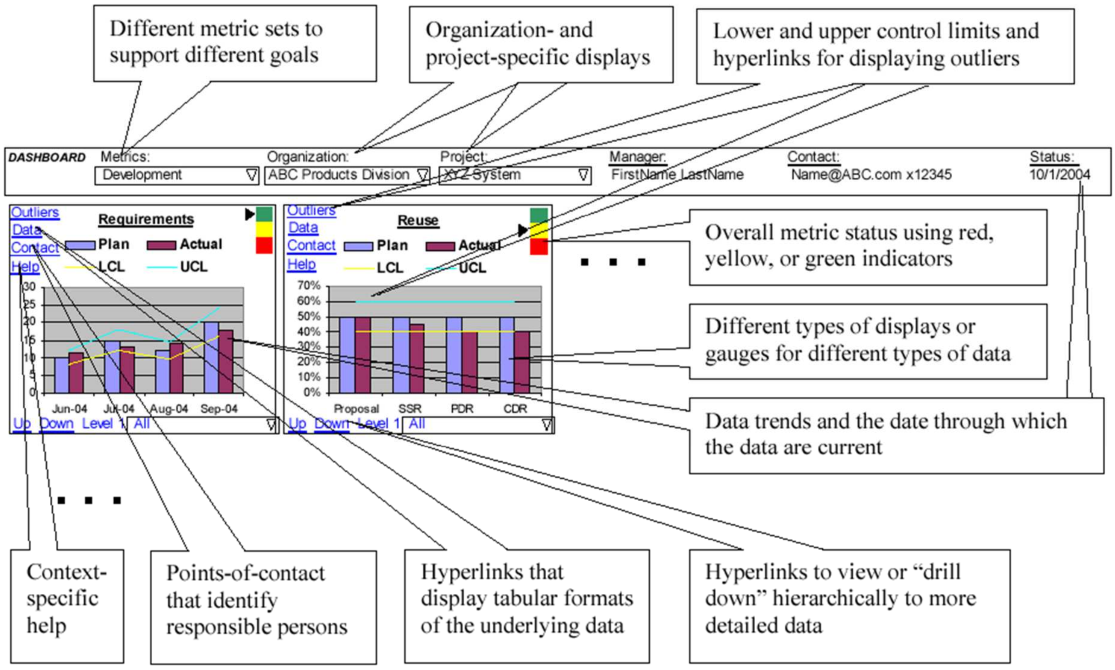
Data accuracy

- Don't collect unnecessary data
  - The questions to be answered should be decided in advance and the required data identified
- Tell people why the data is being collected
  - It should not be part of personnel evaluation
- Don't rely on memory
  - Collect data when it is generated not after a project has finished

Data presentation

- Reports
- Web reports
- Dashboard

Dashboard



Personnel

Project personnel

- Key activities requiring personnel:
  - requirements analysis
  - system design
  - program design
  - program implementation
  - testing
  - training
  - maintenance
  - quality assurance

Choosing personnel

- ability to perform work
- interest in work
- experience with
  - similar applications
  - similar tools or languages
  - similar techniques
  - similar development environments
- training
- ability to communicate with others
- ability to share responsibility
- management skills

Work styles

- Extroverts: tell their thoughts
- Introverts: ask for suggestions
- Intuitives: base decisions on feelings
- Rationals: base decisions on facts, options

Organizational structure

- Depends on
  - backgrounds and work styles of team members
  - number of people on team
    - n people, max interactions = $n^2/2$
  - management styles of customers and developers
- Examples:
  - Chief programmer team
  - Egoless approach

Organizational structures

Highly structured

- high certainty
- repetition
- large project

Loosely structured

- uncertainty
- new technology
- small projects

Risk management

Risk management

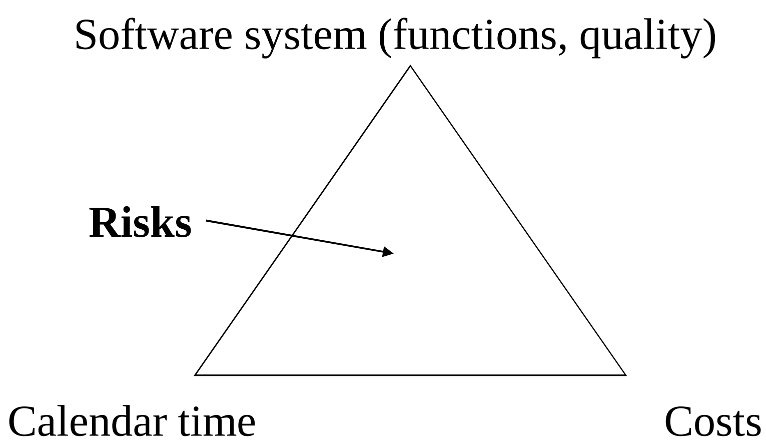
- Project Management for adults

If you don't actively attack the risks,

they will actively attack you

Tom Gilb

Risk Management



Strategies

- Reactive
  - “Indiana Jones school of risk management”
  - Risk management = Crisis management (“fire-fighting mode”)
- Proactive

Risk management (proactive)

- Identify risks
- analyze them
- quantify effects
- define strategies and plans to handle them

Risk

- Future event that can have (bad) impact on project

Risk categories

- Project
- Technical
- Business

- Known
- Predictable
- Unknown

Project Risks

- Regarding (ill defined) project plan
  - budget, personnel, timings, resources, customers
- Regarding management
  - No management support
  - Missing budget or people

Technical risks

- Regard fesibility of product
  - Design, interfaces, verification, ..

Business risks

- Regarding market or company
  - No market for the product (*market risk*)
  - Product not in scope with company plans (*strategic risk*)
  - Sales force does not know how to sell the product (*sales risk*)

Known risks

- Identified before/during risk management
- Ex:
  - Unrealistic deadlines
  - No requirements
  - No focus
  - Poor development environment

Predictable risks

- From previous experience
- Ex.
  - Personnel turnover
  - Poor communication with customer

Boehm's top ten risk items

- Personnel shortfalls
- Unrealistic schedules and budgets
- Developing the wrong functions
- Developing the wrong user interfaces
- Gold-plating
- Continuing stream of requirements changes
- Shortfalls in externally-performed tasks
- Shortfalls in externally-furnished components
- Real-time performance shortfalls
- Straining computer science capabilities

Other common risks

- instability of COTS (Commercial Off-The-Shelf) components/products
- interface with legacy
- stability of development platform (hw + sw)
- limitations of platform
- multi-site development
- use of new methodologies / technologies
- standards, laws
- development team involved in other activities
- communication/language problems

Risk management terms

- Risk impact: the loss associated with the event
- Risk probability: the likelihood that the event will occur
- Risk control: the degree to which we can change the outcome

Risk exposure = (risk probability) x (risk impact)

RM Process

- 1- Risk assessment
  - identification
  - analysis
  - ranking
- 2- Risk control
  - planning
  - monitoring

Identification

- identify risks
  - checklist, taxonomies, questionnaires
    - PMI (Project Management Institute, PMBOK)
    - SEI (SEI-93-TR-06)
      - ex: technical, management, business risks
  - brainstorming
  - experience

Analysis

- probability
  - very high, high, medium, low, very low
- impact
  - catastrophic, critical, marginal, negligible
- exposure
  - probability * impact

Exposure

Impact/ probability	**Very high**	**High**	**Medium**	**Low**	**Very low**
Catastrophic	High	High	Moderate	Moderate	Low
Critical	High	High	Moderate	Low	Null
Marginal	Moderate	Moderate	Low	Null	Null
Negligible	Moderate	Low	Low	Null	Null

Ranking

- By exposure
- by qualitative assessments
  - only higher exposure risks are handled

RM Process

- 1- Risk assessment
  - identification
  - analysis
  - ranking
- 2- Risk control
  - planning
  - monitoring

Planning

- For selected risks (high in exposure)
  - define corrective actions
  - evaluate cost, decide if acceptable
  - insert actions in project plan

Three strategies for risk reduction

- avoiding the risk: change requirements for performance or functionality
- transferring the risk: transfer to other system, or buy insurance
- assuming the risk: accept and control it

risk leverage = difference in risk exposure divided by cost of reducing the risk

Ex.

- ABS for car, software controlled. More flexible, but risk of failure from software
  - Avoiding. No software controlled
  - Transfer. Insurance.
  - Assuming. Develop software with best techniques, apply redundancy.

Ex.

- Risk leverage
  - ABS, software developed normally
    - cost 100KEuro,
    - risk exposure = 10^{-3} * 1M Euro
  - ABS, software developed best techniques
    - cost 1M Euro,
    - risk exposure = 10^{-6} * 1M Euro
  - Risk leverage

    10^{-3} * 1M Euro – 10^{-6} * 1M Euro / (1M – 100k)Euro

Company profiles and risk handling styles

- project owner – takes charge of risk
- fixed price contract
- work provider – no interest in risk

SoftEng http://softeng.polito.it

Monitoring

- follow project plan, including corrective actions
- monitor status of risks
- identify new risks, assess them, update ranking

SoftEng http://softeng.polito.it

Monitoring (2)

- Is part of PM that has to consider also
  - risk log (document)
  - risk reviews (activities)
    - also with external assessors
    - can be coupled with project reviews

Risk log

Risk	Probability	Impact	Exposure	Action	Status
hw platform not available	high	Critical	high	Add software Layer compatible with other platforms	Under control

SoftEng http://softeng.polito.it

Actions for risks

- Personnel shortfalls
  - hire the best, the most suitable, training, team building, technical reviews
- unrealistic schedules and budget
  - more detailed plans, iterative process, reuse, new plans
- instability of components (COTS)
  - qualification, detailed analysis of product and vendor, software layer.

- inadequate requirements
  - prototyping, JAD, iterative process, include user representative in process
    - Joint Application Development
- inadequate user interface
  - study user needs, usability analysis, prototyping
- requirement changes
  - suitable design, iterative process, rigid change control

- Interface with legacy
  - reengineering, encapsulation, incremental change
- subcontractors
  - contracts and payments, team building, assessments before and during
- new technologies
  - prototyping, cost benefit analysis

References

- www.pmi.org project management institute
- www.sei.cmu.edu
- www.ifpug.org
- http://www.itmpi-journal.com
- www.fhg.iese.de - Fraunhofer IESE
- Rapid Development - Taming Wild Software Schedules, Steve McConnell, Microsoft Press, 1996
- Software Engineering Risk Management, Dale Walter Karolak, IEEE Computer Society Press, 1996
- Assessment and Control of Software Risks , Caper Jones, Yourdon Press, 1994
- Software Risk Management - Principles and Practices, Barry W.Boehm, IEEE Software, Vol 8, No. 1, Jan 1991, PP32-41
- Taxonomy-Based Risk Identification, M.J.Carr et al., CMU/SEI-93-TR-06, SEI, 1993
- Www.riskwatch.com - Risk management tools