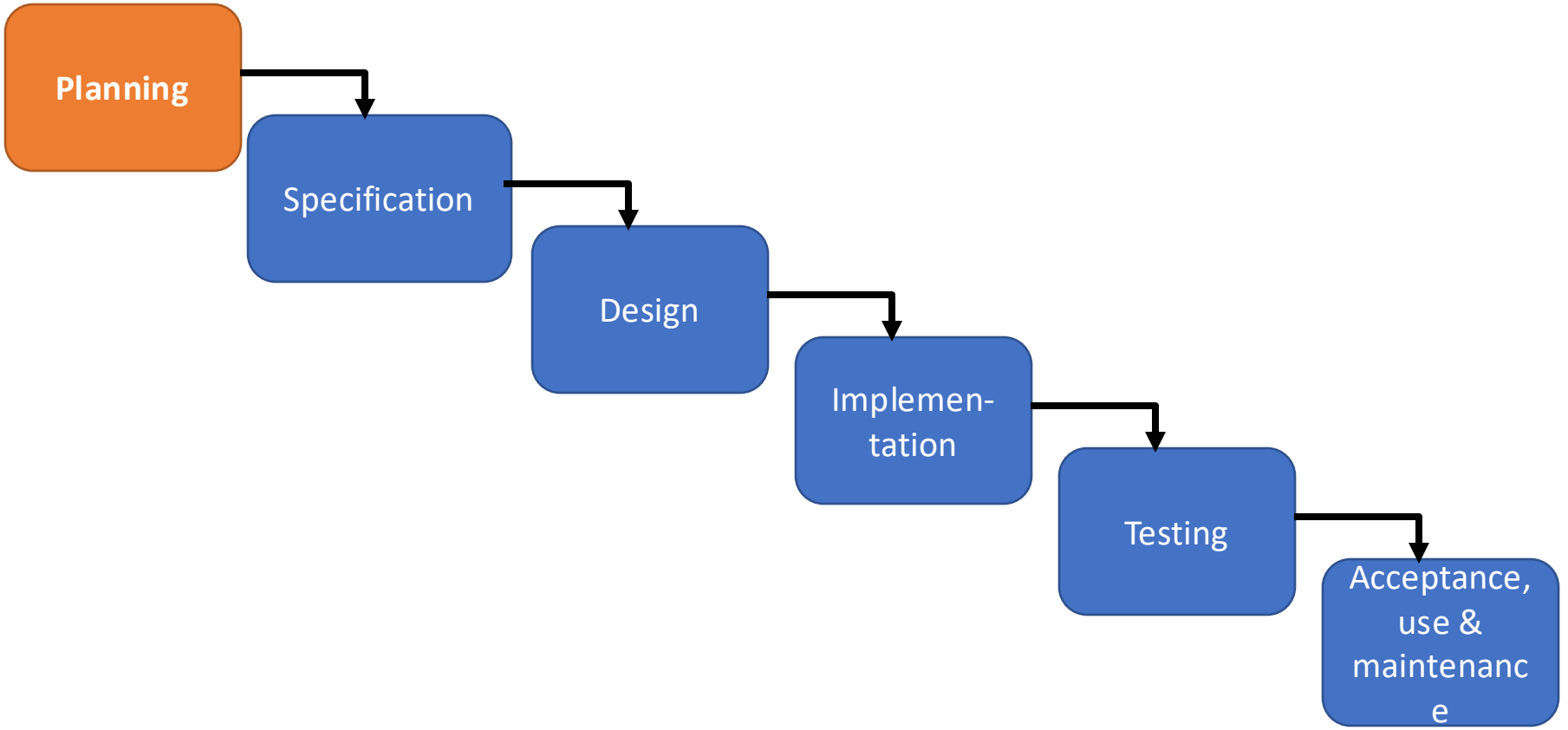
Chapter 7 - Estimation Methods

Walter F. Tichy

Where are we right now?



Learning Objectives

- Be able to reproduce the estimation methods listed and apply them to examples.
- Content
  - Factors influencing effort estimation
  - Basic methods of effort estimation
  - COCOMO II

Effort Estimation

- Effort estimation is one of the least popular activities in software engineering, yet it is critical to the commercial success of a software house.
- Profitability of a product:
  - Profit (loss) = margin * estimated quantity - one-time development costs
  - where
    - margin = price - variable costs

Effort Estimation

- View of the software manufacturer or the contractor:
  - Development costs of a software system:
    - Main share of development costs: personnel costs
    - Computer and software costs (licenses) for product development
    - Costs for other services, training, office supplies, printing costs, documentation, travel expenses, etc.
    - variable costs: marketing, advertisement, maintenance, support, consultants, licenses, office rental

Effort Estimation

- Methods for cost and schedule estimation
  - Most models are based on the estimated size of the software product to be created in number of program lines, or in **Lines of Code** (LOC), or **1000 Lines of Code** (KLOC)
    - For example, for higher level languages, all declaration and statement lines are estimated (without comment lines, empty lines)
    - The estimated LOC is divided by an empirical value for programming productivity (in LOC) of an employee per year or month
    - Result: estimated effort in person-years (PJ) or person-months (PM).
    - 1 PJ = 9 PM or 10 PM.

Effort Estimation

- A **person-month** **(PM)** is the amount of work a person accomplishes on average in a month.
  - A PM consists of 4 person weeks (PW)
  - One PW consists of 5 person days (PD)
  - One PD consists of 8 person hours
- So, in one calendar month one person works
  4 PW/PM * 5 PD/PW * 8 h/PD = 160 h
- With a productivity of approx. 3,500 LOC/PJ and
  10 PM/PJ, this results in **350 LOC/PM**, i.e., approx.
  2.12 LOC/person hour.
- This includes not just coding, but all the other activities, such as requirements elicitation, modeling, specification, design, test, documentation, meetings

Influencing Factors

- The determined effort is divided by the development time available according to the schedule
  - Result: The number of employees to be deployed for the development
- Rules of thumb
  - An average software development delivers about 350 tested source code lines (not counting comment lines) per engineering month
  - This figure appears small. The reason is that it includes the contributing costs of all phases: definition, specification, design, implementation, and test (but not maintenance).

Influencing Factors

- Example
  - A software product with an estimated 21,000 LOC is to be realized
  - Average productivity per person: 3,500 LOC/year
  - 6 person years are required
  - So:
    - If 3 employees work together in a team, then 2 calender years are needed until completion
    - If the product has to be completed within 6 months, 12 employees are required for this purpose

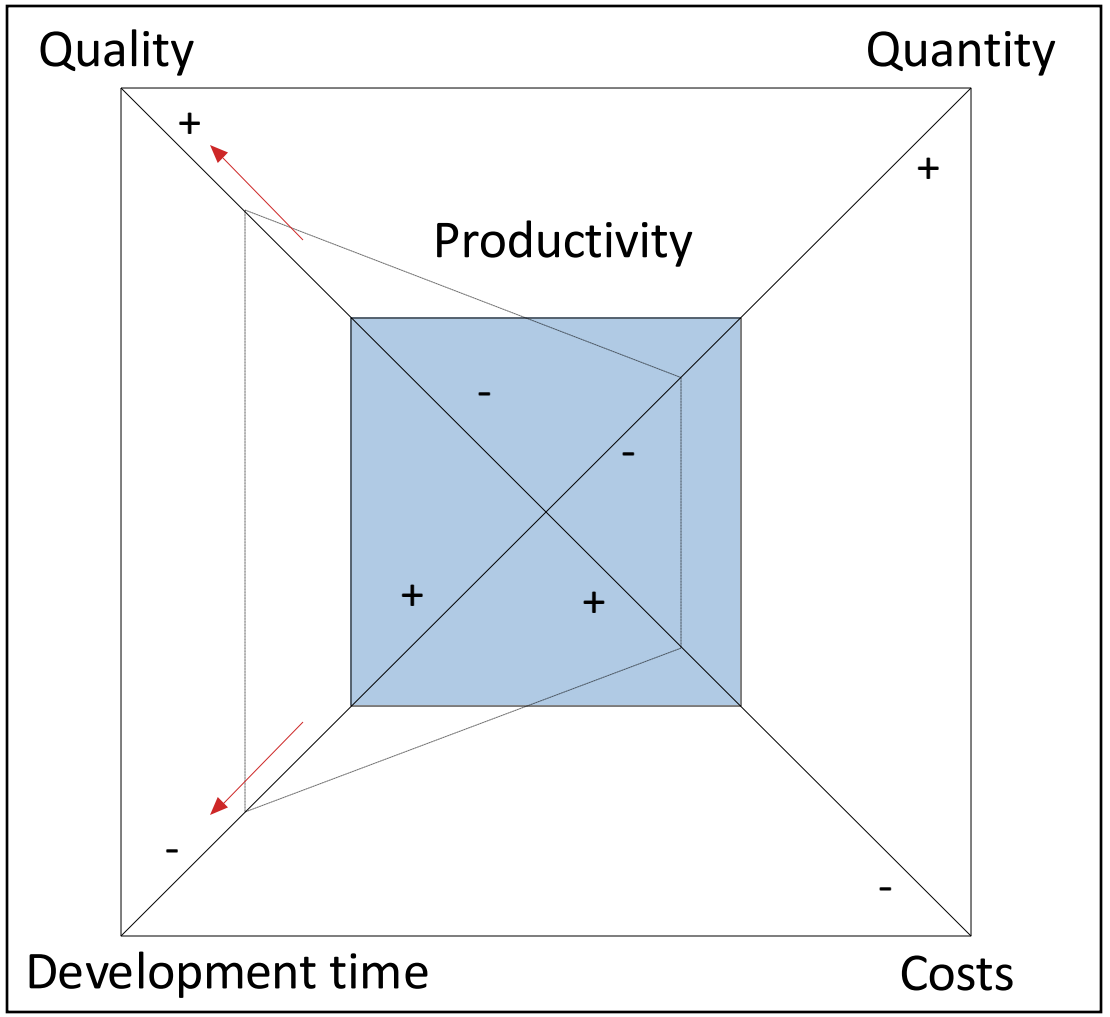
Is the correlation between effort and development time linear?

Influencing Factors

- Important influencing factors
  - Quantity
  - Quality
  - Development time
  - Costs

- Influence each other - illustrated in the “devil's square”

The “Devil's Square”



One can drag the corners of the inner quadrilateral,
but at the same time
its area remains the same

Influencing Factors: Quantity

- Scope
  - Estimate "Number of program lines" (KLOC)
  - linear or disproportionate relationship between KLOC and effort
  - Value depending on programming language
- Complexity
  - qualitative measures ”easy", "medium" and "severe".
  - mapping to a numeric scale
    - Example: Grades between 1 and 6.

Influencing Factors: Quality

- The higher the quality requirements, the greater the effort required
- There is not "the quality", but there are different quality characteristics
- Key figures can be assigned to each quality characteristic

Influencing factors: Development Duration

- If the time is to be shortened, then more employees are needed
- More employees increase the communication within the development team
- more time for communication reduces productivity
- If the development time can be extended, then fewer employees are needed
  - The communication portion decreases
  - The productivity of each employee increases
- The correlation between effort in PM and development time is therefore only approximately linear.

Influencing Factors: Productivity

- Influenced by many different factors
  - Ability to learn
  - Motivation of employees
  - Programming languages, methods and tools used
  - Training/familiarity with application area
  - Work climate

How do you come up with estimated LOC or PM?

With the analogy method:

- Compare the development to be estimated with already completed product developments based on similarity criteria

- Criteria
  - same or similar field of application
  - same or similar product scope
  - Same or similar level of complexity
  - same programming language
  - same programming environment/tools

Analogy method: Example

- Pascal compiler = 20 PM
- Modula compiler = ?
  - 20% new constructs
  - 50% of the code reusable
  - 50% must be revised
- Estimate
  - 50% slightly modified: 10 PM * 0.25 = 2.5 PM
  - 50% complete revision: 10 PM
  - 20% additional new development of high complexity:
    - 4 PM * 1.5 (complexity allowance) = 6 PM
- Modula compiler estimate = 18.5 PM.

Analogy Method

- Advantages:
  - Relatively simple and intuitive estimation method
- Disadvantages:
  - global estimate based on individual experience, not transferable
  - lack of a general approach
- Variant of this: "Planning poker".
  Several estimators give an estimate (initially concealed). Those who are far apart must give reasons; then there is a discussion. Then a new round of poker follows until an agreement is reached (there are even playing cards for this; see later)

The COCOMO II estimation method

- Calculates the total duration of a SW project in person-months from the estimated size and 22 influencing factors.
- The size is estimated in KLOC or alternatively in unadjusted function points (identify individual functions visible on the user interface, as well as I/O operations and assign point values to them)
- COCOMO II is the successor to the first COCOMO developed in 1981.
  - COCOMO: *Constructive Cost Model*

Estimation formula of COCOMO II

$$PM = A \cdot (Size)^{1,01+0,01 \cdot \sum_{j=1}^{5} SF_j} \cdot \prod_{i=1}^{17} EM_i$$

- *PM*: Number of person months
- *A*: Constant for the calibration of the model (e.g., for LOC or function points).
- *Size*: estimated size of the software in KLOC or unadjusted function points.
- SF_j : *scale factors*
- Due to the exponent > 1, the effort in PM grows somewhat disproportionately
- EM_i : *multiplicative cost drivers*

COCOMO II scaling factors

Scale Factors (SF_i)	Very Low (5)	Low (4)	Nominal (3)	High (2)	Very High (1)	Extra High (0)
Precedentedness	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
Development Flexibility	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
Architecture / risk resolution	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
Team cohesion	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
Process maturity	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

COCOMO II Multiplicative cost factors

- Product factors
  - Required Software Reliability, Data Base Size, Product Complexity, Required Reusability, Documentation match to life-cycle needs
- Platform factors
  - Execution Time Constraint, Main Storage Constraint, Platform Volatility, Computer Turnaround Time
- Personnel factors
  - Analyst Capability, Programmer Capability, Applications Experience, Platform Experience, Language and Tool Experience, Personnel Continuity
- Project factors
  - Use of Modern Programming Practices, Use of Software Tools, Multisite Development, Required Development Schedule, Classified Security Application
- A nominal value of 1 is preset for each of the 17 factors.

COCOMO II - Example calculation (1)

- Assumptions:
  - A has the value 3
  - All cost factors have the value 1 (nominal value)
  - The scaling factors are:
    - SF(Very Low): All SF have the value 5: B = 1,26
    - SF(Nominal): All SF have the value 3: B = 1,16
    - SF(Extra High): All SF have the value 0: B = 1,01
  - The size of the projects ranges from 10000 LOC (10 KLOC) to 200000 (200 KLOC).

$$\prod_{i=1}^{17} EM_i = 1$$

$$B = 1{,}01 + 0{,}01 \cdot \sum_{j=1}^{5} SF_j$$

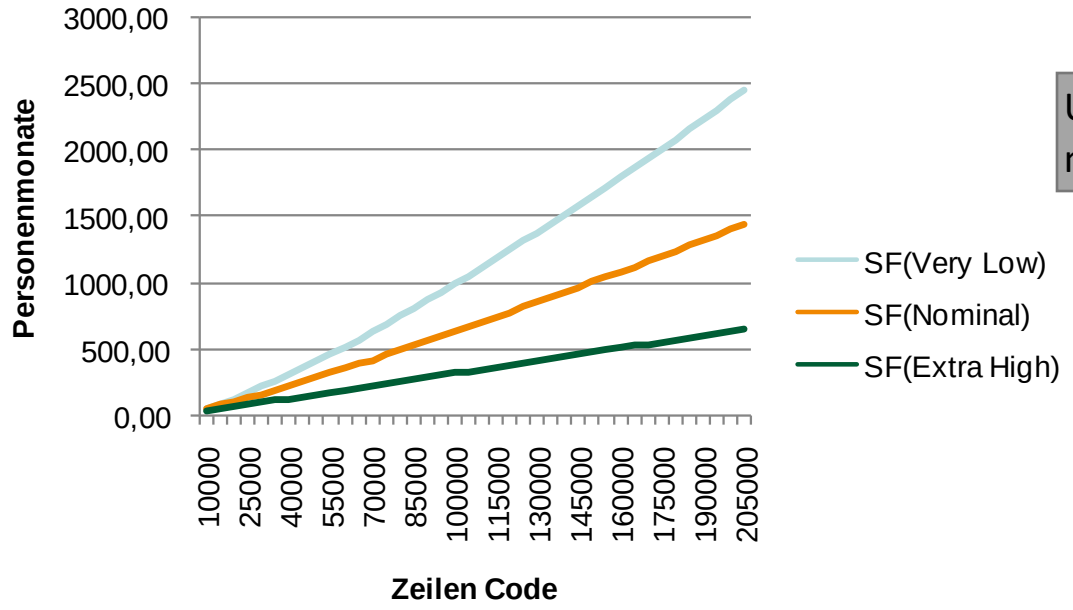
$$PM = A \cdot (Size)^{1{,}01+0{,}01 \cdot \sum_{j=1}^{5} SF_j} \cdot \prod_{i=1}^{17} EM_i$$

$$PM = 3 \cdot (Size)^{B} \cdot 1 = 3 \cdot (Size)^{B}$$

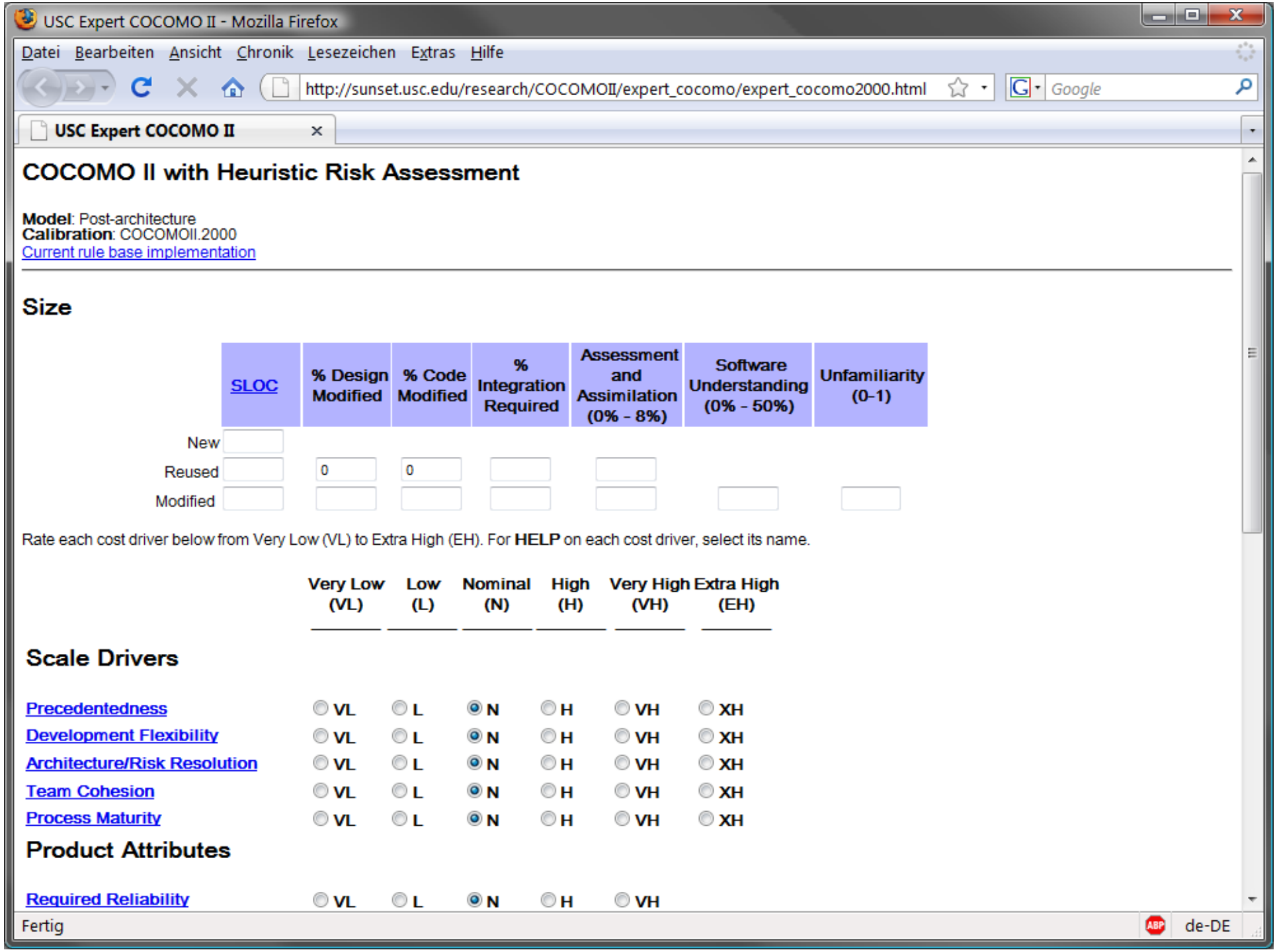
COCOMO II - Example calculation (2)

$$PM = 3 \cdot (Size)^{B}$$

	10000	15000	20000	25000	30000	...	205000
SF(Very Low)	54,59	90,99	130,74	173,19	217,92	...	2454,32
SF(Nominal)	43,36	69,40	96,90	125,53	155,09	...	1441,30
SF(Extra High)	30,70	46,24	61,82	77,45	93,11	...	648,62



Unit: person months


USC Expert COCOMO II - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://sunset.usc.edu/research/COCOMOII/expert_cocomo/expert_cocomo2000.html
Google
USC Expert COCOMO II
COCOMO II with Heuristic Risk Assessment
Model: Post-architecture
Calibration: COCOMOII.2000
Current rule base implementation
Size
SLOC
% Design Modified
% Code Modified
% Integration Required
Assessment and Assimilation (0% - 8%)
Software Understanding (0% - 50%)
Unfamiliarity (0-1)
New
Reused
Modified
0
0
Rate each cost driver below from Very Low (VL) to Extra High (EH). For HELP on each cost driver, select its name.
Very Low (VL)
Low (L)
Nominal (N)
High (H)
Very High (VH)
Extra High (EH)
Scale Drivers
Precedentedness
Development Flexibility
Architecture/Risk Resolution
Team Cohesion
Process Maturity
VL L N H VH XH
Product Attributes
Required Reliability
VL L N H VH
Fertig
de-DE

Consensus Estimation Methods

- Based on the estimates of "experts".
  - Delphi Method
  - *Planning poker*

Delphi estimation method

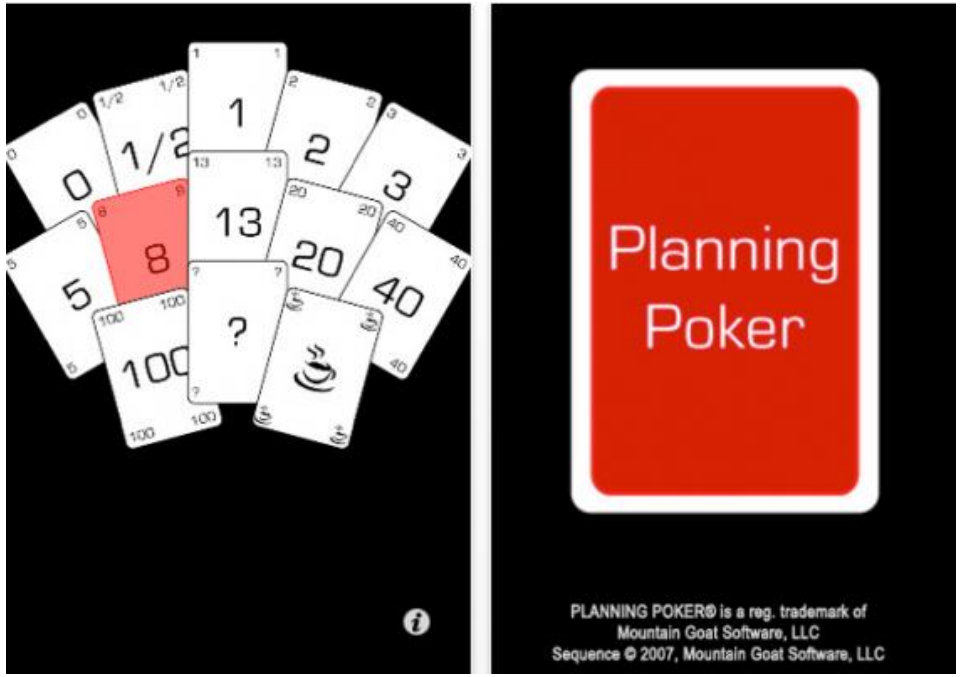
- One uses a set of estimators (experts) who have experience with the planned SW.
- In one or more rounds, the following is done:
  - Each estimator anonymously provides an estimate plus rationale on a card.
  - The moderator summarizes the results, including the justifications
  - If the values are far apart, a new round is conducted in which the estimators are allowed to change their estimate. The hope is that the estimated values will converge towards the "correct" value.
- If nothing changes, take the average value.
- Important: first estimate is not influenced by other participants.

Planning Poker:
Variation of the Delphi method

- All participants are given a deck of cards with rapidly increasing values, e.g., 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞, and ? ("impossible to guess")
- Each participant lays a card with an estimate on the table, with the number facing down. (Unit of estimation can be hours, days, or months; will be agreed beforehand).
- After that, everyone reveals their cards at the same time.
- The estimators with high and low values justify their estimate.
- Process is repeated until consensus is reached.
- An timer keeps the process short.

Planning poker (2)

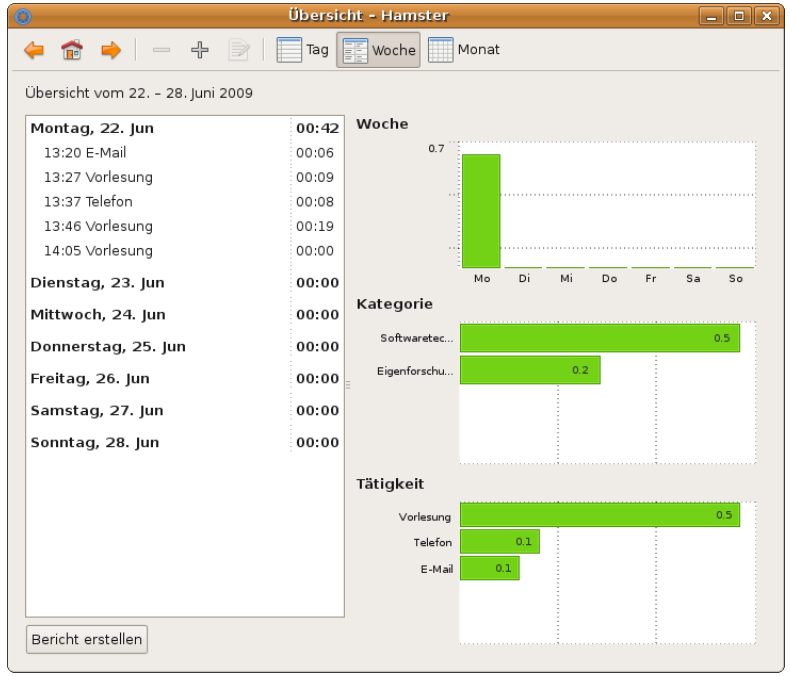
Of course there are several mobile applications….



And what to use now?

- Depending on the time of the estimate and knowledge of effort-relevant data, one or the other method should be used:
- For early, rough estimates use the
  - Analogy method
  - Planning poker
- When development has stared, size estimates are possible, and the influencing factors are known, use
  - COCOMO II
- In any case: collect productivity data (LOC per month or per year per person) . Keep a log of what you're doing! That way, at least you know your own productivity.

Keeping Track of your Time



Hamster

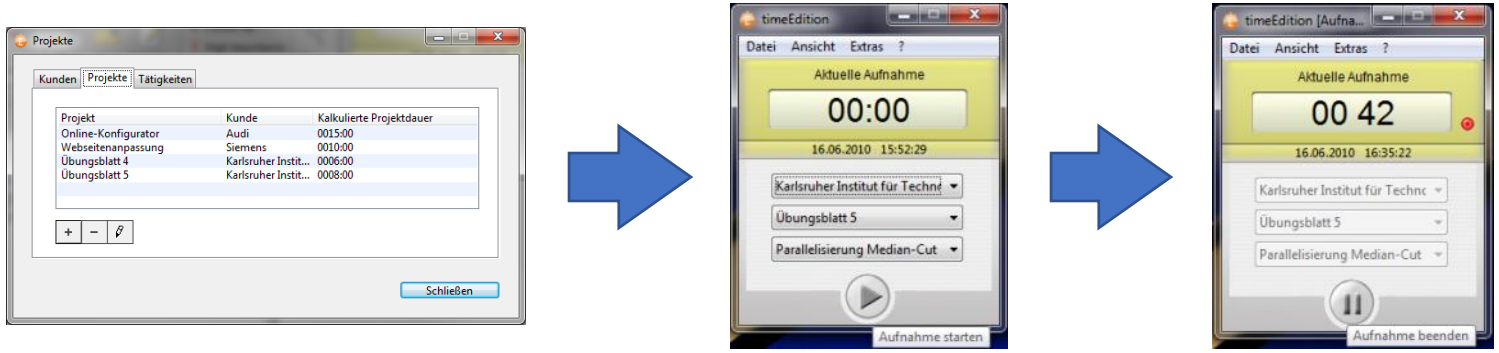
for Linux

timeEdition

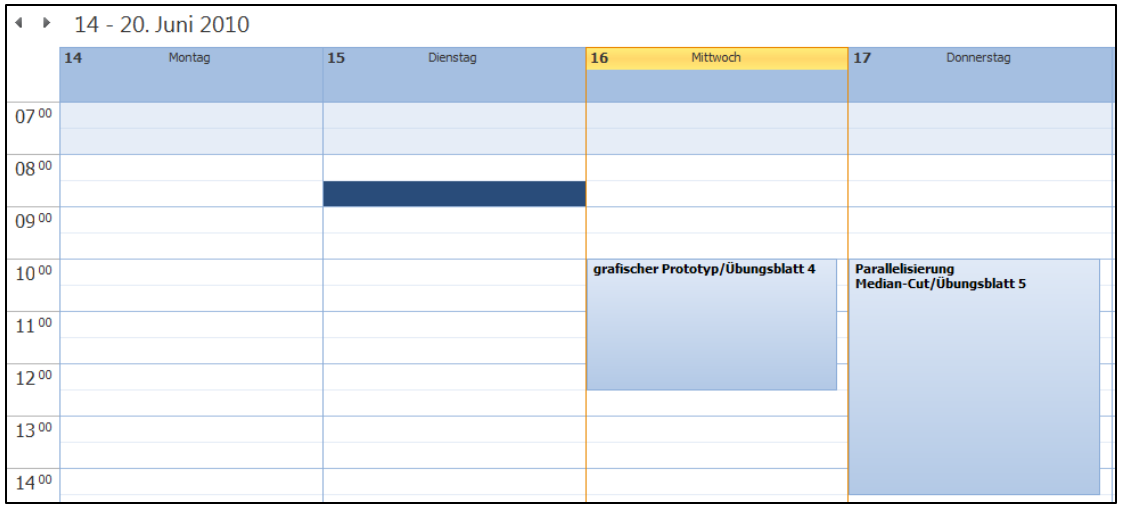
für Windows, Mac, Linux

Keeping Track of your Time

- Projects and starting/stopping timer for all your activities



- Import/export, for planning, billing, self-optimization, time estimation



timeEdition is not free.

Search for “free time tracker”.