Politecnico
di Milano

Project Management

Credits to:
Hans van Vliet
Moreno Marzolla,
Ian Sommerville
Damian Tamburri

Roadmap

- Software Project Management
- Project Planning
  - Size, effort and cost estimation
- Risks and People Management
- Project and Process Improvement: CMMI

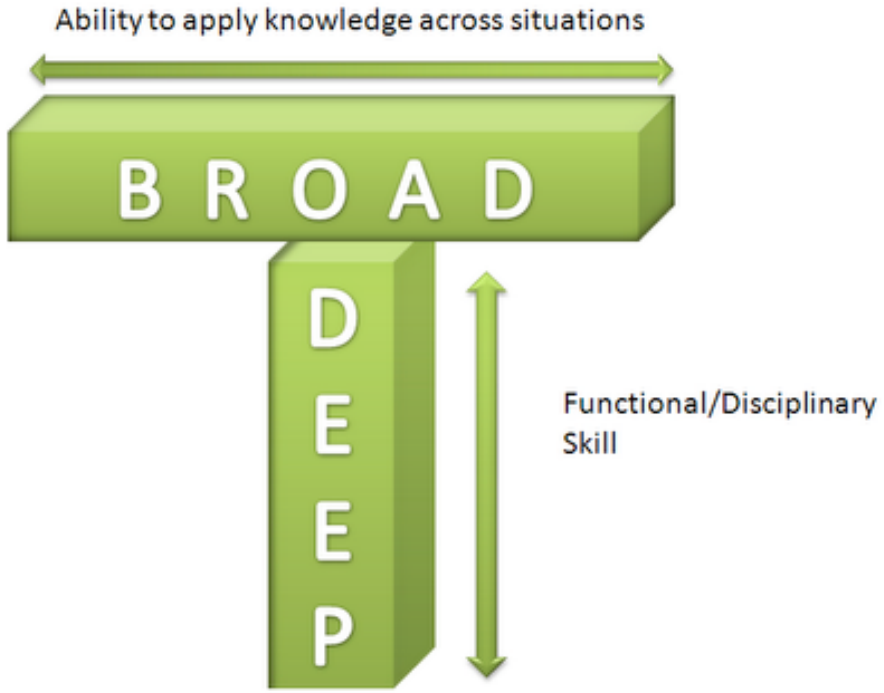
Who is the project manager?

The project in this case has an intriguing name: “cosa-nostra”


HOW LONG WILL IT TAKE TO FIX THE BUGS IN OUR CONTROL MANAGEMENT SOFT-WARE?
DO YOU WANT A REALISTIC ESTIMATE THAT WILL RUIN YOUR DAY, OR A LIE THAT WILL ALLOW YOUR IGNORANCE AND YOUR HAPPINESS TO LOCK ARMS AND SQUARE DANCE TO THE NEXT CUBICLE?
THAT SECOND OPTION SOUNDS FESTIVE.
I'M A PLEASER.
Dilbert.com DilbertCartoonist@gmail.com
© 2010 Scott Adams, Inc./Dist. by UFS, Inc.

Who is the project manager?

- An experienced and **very bold** professional
- More than **just technical** experience
- T-Shaped Professional figure



Who is the project manager?

- But also...
  - A **stressed** professional
  - A **creative** professional
  - Almost equally good at **writing and communicating** than at **managing people and things**

Software Project Management: what is a (Software) Project?

- A project is a planned set of activities that
  - Has a **starting** date
  - Has an **estimated** ending date
  - Has a **single** objective
  - Is realized by **one or more** teams
  - Exploits a more or less **flexible** set of resources

Software Project Management

- Activities involved in ensuring that software is delivered on time and on schedule and in accordance with context and requirements
- Requires the interaction of ***economical***, ***social, technical*** *and* ***organizational*** aspects
- A well-directed project ***may*** fail, a badly-directed project ***certainly*** fails
- Experience with previous projects is important and has to be considered a company-wide memory, i.e., organizational culture
- It is difficult to teach how to be a good manager… experience certainly helps!

Software Project Management (2)

- *Project planning*
  - Project managers are responsible for planning: e.g, estimating and scheduling project development, assigning people to tasks.
- *Reporting*
  - Project managers are usually responsible for reporting: progress of a project; progress to customers; progress to business IT managers.
- *Risk management*
  - Project managers assess and monitor risks that may affect a project, taking action if needed.

Software Project Management (3)

- *People management*
  - Project managers choose people for their team and establish ways of working for effective team performance
- Addendum Activity: *proposal writing*
  - The first stage in a software project may involve writing a proposal to win a contract.
  - The proposal describes the objectives of the project and how it will be carried out.

Software Project Management (5): Delay and Failure

- Unrealistic deadlines, e.g., it is imposed by someone external to the technical staff
- Requirements change (too) often
- Effort and resources have been estimated in an overly optimistic way,
- Risks have not been taken into account from the start of the project.
  - Risks can be technical or human difficulties
- Communication problems among the staff members
- Difficulty by the management to recognize recurrent delays and take immediate action
- Subversive stakeholders

Software Project Management (5): Delay and Failure… **An Example?**

- **Unrealistic deadlines, e.g., it is imposed by someone external to the technical staff**
- **Requirements change (too) often**
- Effort and resources have been estimated in an overly optimistic way,
- Risks have not been taken into account from the start of the project.
  - Risks can be technical or human difficulties
- Communication problems among the staff members
- Difficulty by the management to recognize recurrent delays and take immediate action
- Subversive stakeholders

Software Project Management (5): Delay and Failure… **An Example?**

- **Unrealistic deadlines, e.g., it is imposed by someone external to the technical staff**
- **Requirements change (too) often**

Software Project Management (5): Delay and Failure… **An Example?**

- **Too many undiscovered stakeholders were eventually reported**
- **Unrealistic deadlines were eventually reported by the staff**

Software Project Management (5): Delay and Failure… **An Example?**

COST: 174,000,000 $ (give or take)

http://www.cio.com/article/2380827/developer/6-software-development-lessons-from-healthcare-gov-s-failed-launch.html

Software Project Management (5): Delay and Failure… **Yet Another Example?**

- Unrealistic deadlines, e.g., it is imposed by someone external to the technical staff
- Requirements change (too) often
- **Effort and resources have been estimated in an overly optimistic way,**
- **Risks have not been taken into account from the start of the project.**
  - **Risks can be technical or human difficulties**
- Communication problems among the staff members
- Difficulty by the management to recognize recurrent delays and take immediate action
- Subversive stakeholders

Software Project Management (5): Delay and Failure… **Yet Another Example?**

- **"Air India Dreamliner flight diverted after software problems" – Feb. 2014**
  - → **Miscommunicated risks**
  - → **Miscommunicating stakeholders**
  - → **Unknown system interaction patterns**

Result: two Grounded Test Flights…*

- Entire project cost... 167 Ml $, about 25% of which is software-related

*http://www.advfn.com/nyse/StockNews.asp?stocknews=BA&article=60949432
http://www.nytimes.com/2014/02/07/business/after-boeing-787-is-diverted-air-india-looks-into-software-problem.html?_r=0

Software Project Management (5): Delay and Failure… **And Another?**

- Unrealistic deadlines, e.g., it is imposed by someone external to the technical staff
- Requirements change (too) often
- Effort and resources have been estimated in an overly optimistic way,
- Risks have not been taken into account from the start of the project.
  - Risks can be technical or human difficulties
- **Communication problems among the staff members**
- **Difficulty by the management to recognize recurrent delays and take immediate action**
- **Subversive stakeholders**

Software Project Management (5): Delay and Failure… **And Another?**

- **Nokia is acquired by Microsoft...**
- **A lot of developers do not want this...**
- **So... They leave!**

- **Those who remain become uncooperative with new partners ➔ *Subversive behavior happens!***

Consequence? Nokia is virtually out of the cells. Market ☹

A scenario example (1)

- Assume that we have to develop some software within 9 months
- After a while, through an accurate analysis and estimation of risks, you realize that it requires at least 14 months
- What do you do?

A scenario example (2)

- **A possible alternative**

  Allocate new budget and include additional people to match allotted timeline

A scenario example (2)

- **A possible alternative**

  Allocate new budget and include additional people to match allotted timeline

Any guesses about this?

A scenario example (2)

- **A possible alternative**

  Allocate new budget and include additional people to match allotted timeline

WRONG ☺


ALICE, I'M ADDING JEFF TO YOUR PROJECT TEAM.
THAT'S LIKE TRYING TO PUT OUT A FOREST FIRE BY DROPPING A BABY ON IT.
I'M AVAIL-ABLE TO HELP, TOO.
OKAY, YOUR JOB IS TO KEEP JEFF FROM DOING ANYTHING.
Dilbert.com DilbertCartoonist@gmail.com

A scenario example (2)

- **A possible alternative**

  Allocate new budget and include additional people to match allotted timeline

WRONG ☺ - Brooke's Law:

"adding manpower to a late software project makes it later."

Brooke's Law, a chorollary:

"9 women can't make a baby in 1 month."

A scenario example (3)

- **A reasonable alternative**
- Plan for two releases, the first one after 9 months with a limited set of functionalities
  - … The rest will be eventually completed, e.g., within 14 months

A scenario example: Lessons Learned

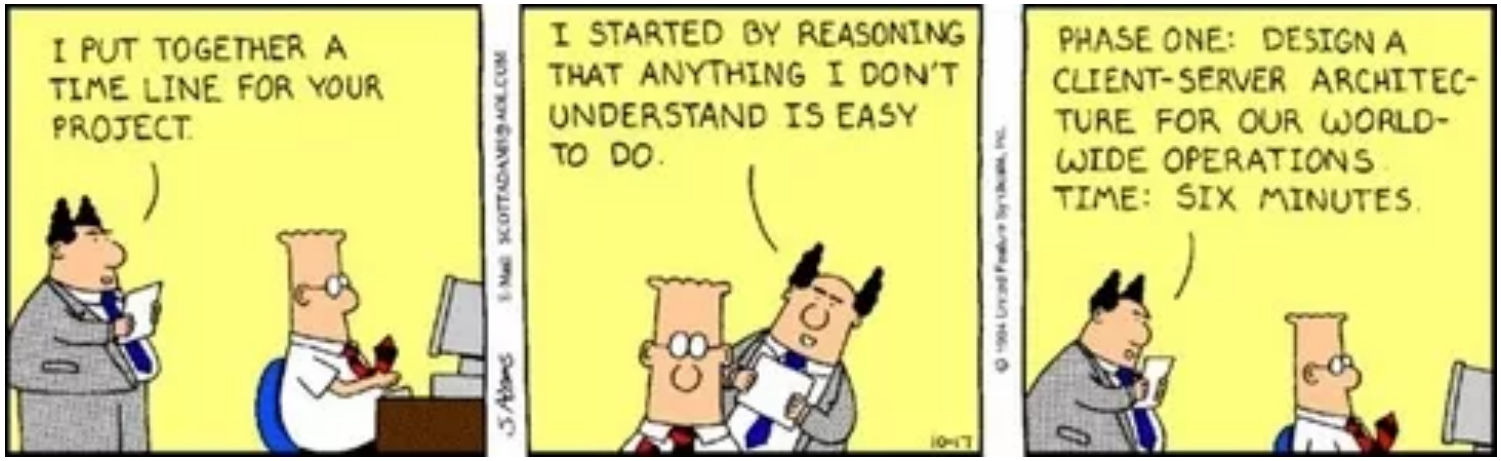
- Software Project Management needs **flexibility and adaptation**
- **Adapting means software and people**
- Management in Sw.Eng. is **as much a work of improvisation than precision**
- Sometimes (or actually most of the times) **products may well have to be ``good-enough''**

Roadmap

- Software Project Management
- **Project Planning**
  - Size, effort and cost estimation
- Risks and People Management
- Project and Process Improvement: CMMI

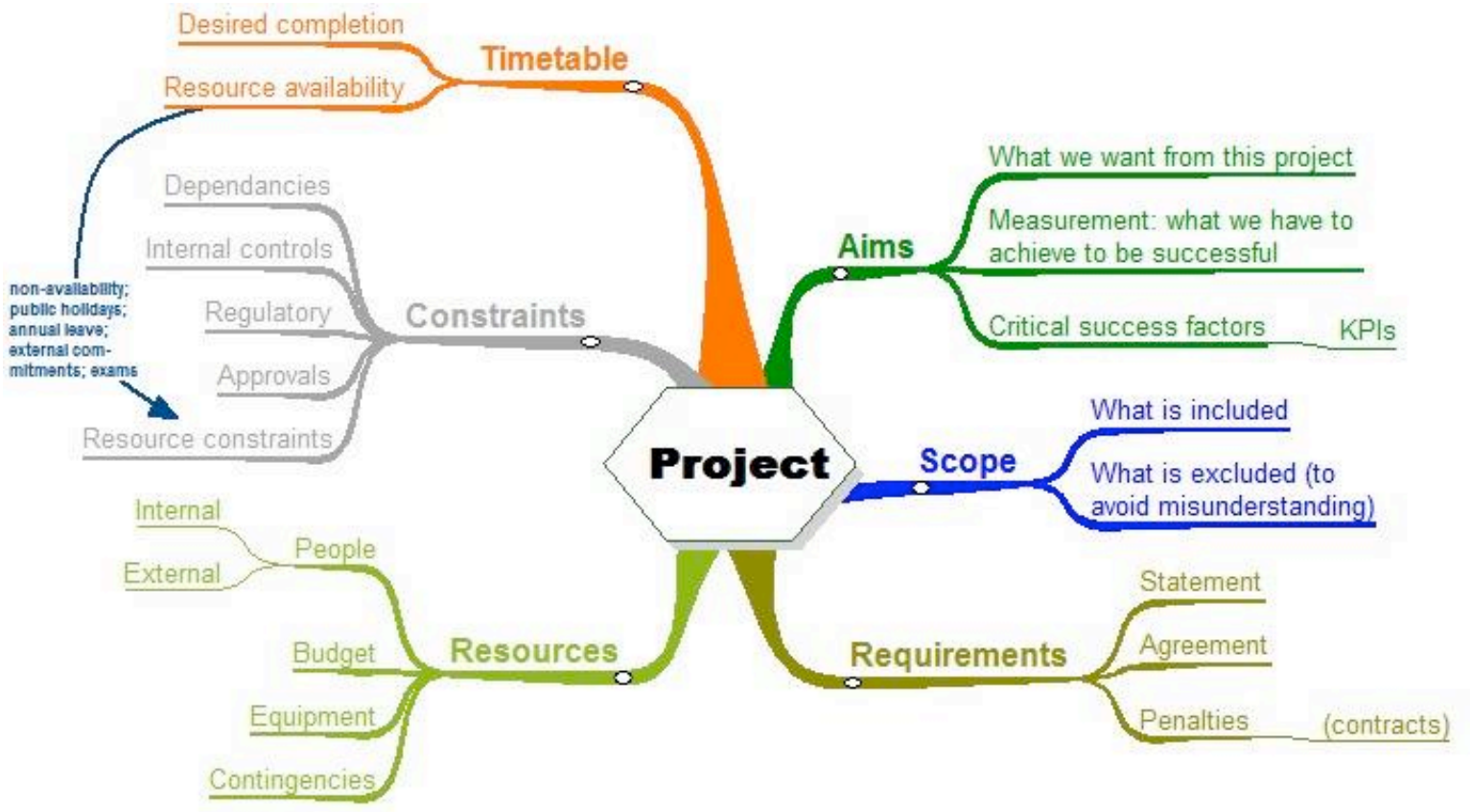
The planning process

- Project planning is an iterative process that starts when you create an initial project plan during the project startup phase.
- Plan changes are inevitable.
  - As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes.
  - Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.

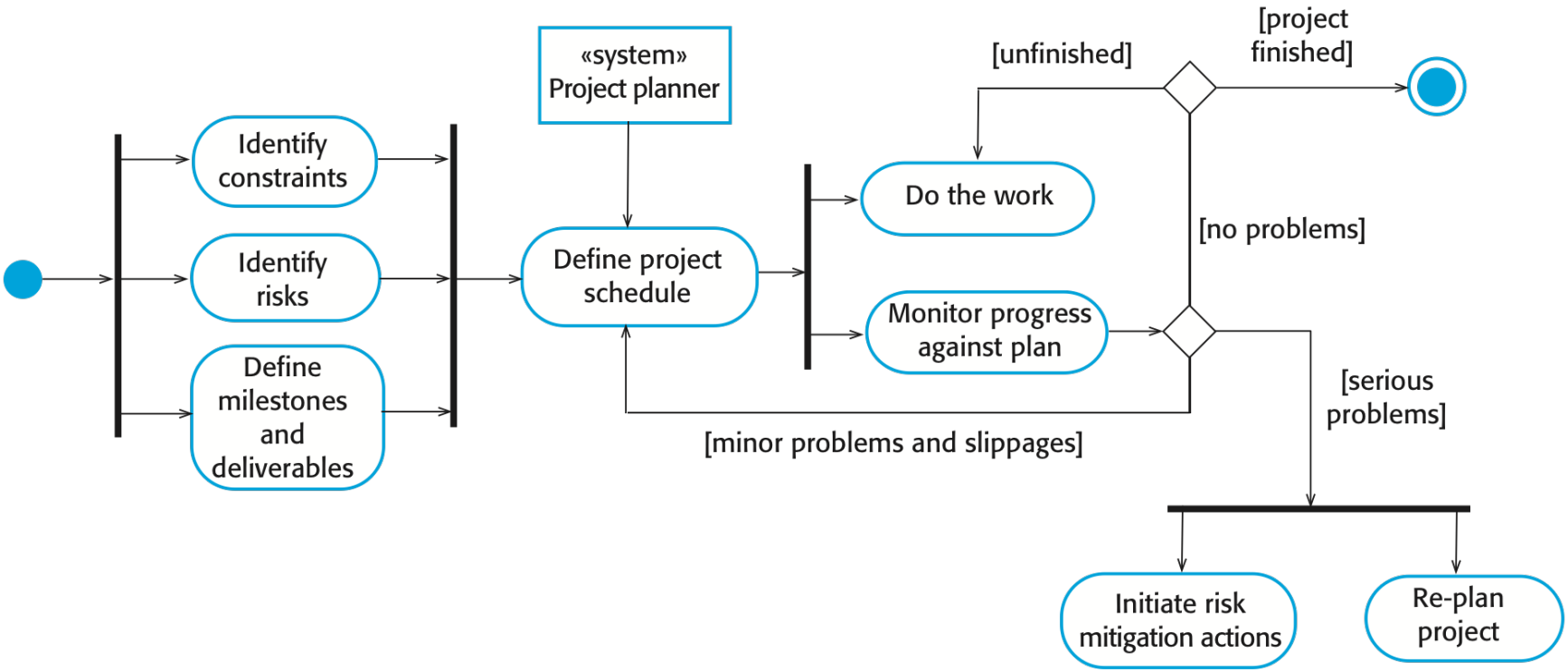

I PUT TOGETHER A TIME LINE FOR YOUR PROJECT.
I STARTED BY REASONING THAT ANYTHING I DON'T UNDERSTAND IS EASY TO DO.
PHASE ONE: DESIGN A CLIENT-SERVER ARCHITECTURE FOR OUR WORLD-WIDE OPERATIONS. TIME: SIX MINUTES.

Project planning: A Mind-Map

Politecnico di Milano



The project planning process



Terminology

- Tasks: activities which must be completed to achieve the project goal
- Milestones: are points in the schedule against which you can assess progress, for example, the handover of the system for testing.
- Deliverables: are work products that are delivered to the customer, e.g. a requirements document for the system.

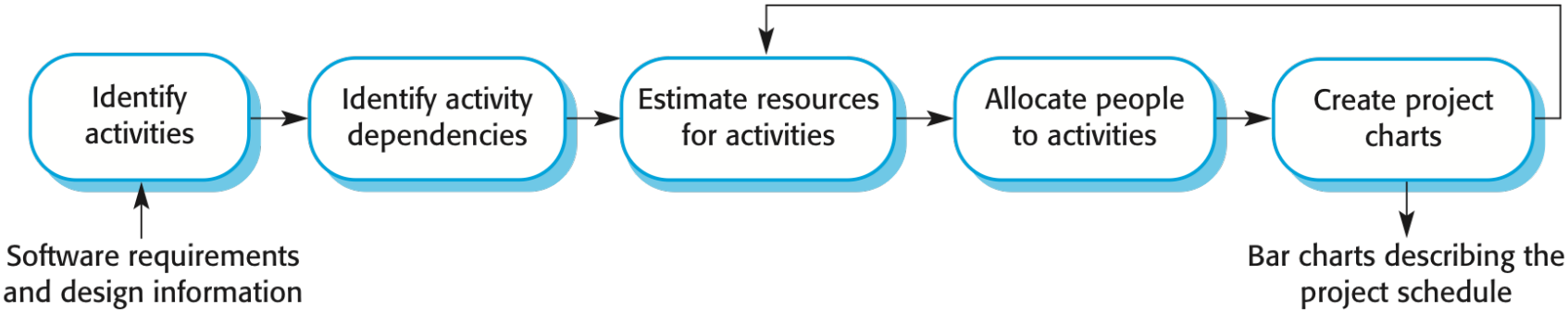
Project scheduling

- Project scheduling
  - deciding how the work in a project will be organized as separate tasks,
  - when and how these tasks will be executed.
- Estimates for
  - the calendar time needed to complete each task,
  - the effort required
  - who will work on the tasks that have been identified.
  - estimate the resources needed to complete each task (e.g., disk or server space, time required on specialized hardware, simulators, travel budget)

Project scheduling activities

- Split project into tasks and estimate time and resources required to complete each task.
- Organize tasks concurrently to make optimal use of workforce.
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Dependent on project managers intuition and experience.

The project scheduling process



Scheduling problems

- Estimating the difficulty of problems and hence the cost of developing a solution is hard (see later for some approaches).
- Productivity is not proportional to the number of people working on a task.
- Adding people to a late project makes it later because of communication overheads.
- The unexpected always happens. Always allow contingency in planning.

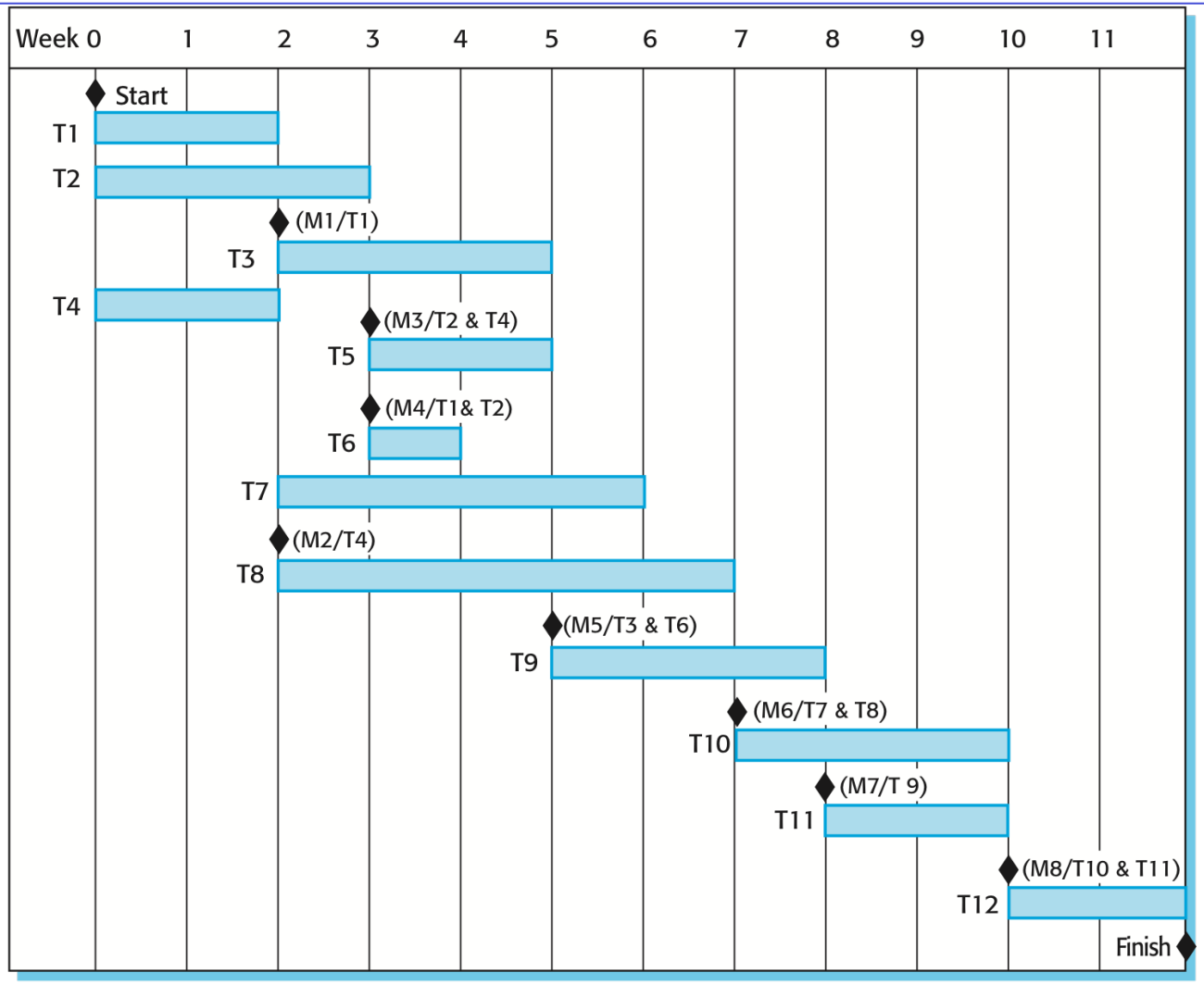
Schedule representation

- Graphical notations are normally used to illustrate the project schedule.
- These show the project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- Bar charts are the most commonly used representation for project schedules. They show the schedule as activities or resources against time.

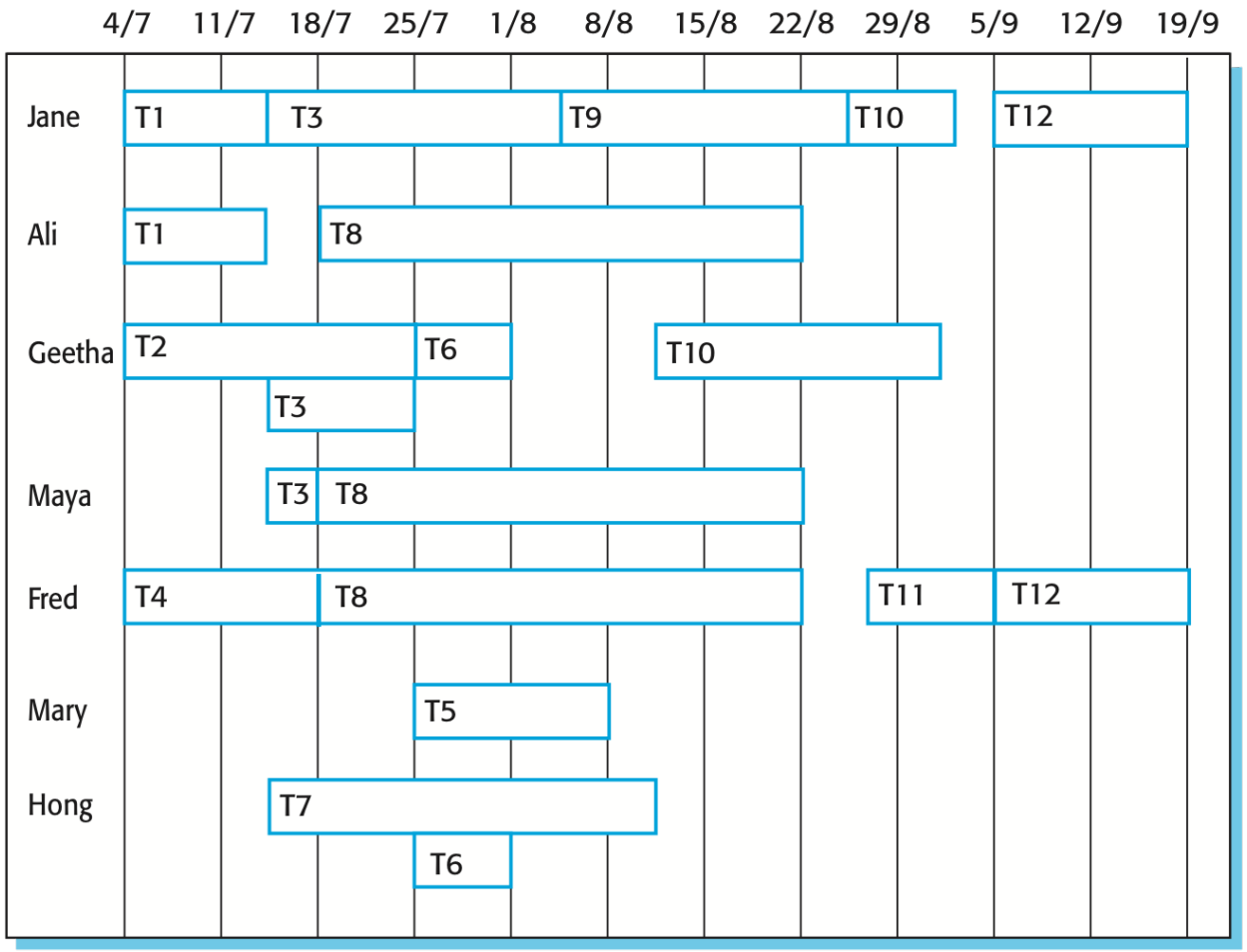
Tasks, durations, and dependencies

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Activity bar chart



Staff allocation chart



Roadmap

- Software Project Management
- Project Planning
  - **Size, effort and cost estimation**
- Risks and People Management
- Project and Process Improvement: CMMI

Thanks for your endurance…

- Break?

Coming up next… Cost Estimation Models!

Software cost: components

- Hardware and software costs
- Travel and training costs
- Effort costs (the dominant factor in most projects)
  - The salaries of engineers involved in the project;
  - Social and insurance costs.
- Effort costs must take overheads into account
  - Costs of building, heating, lighting.
  - Costs of networking and communications.
  - Costs of shared facilities (e.g., library, staff restaurant, etc.

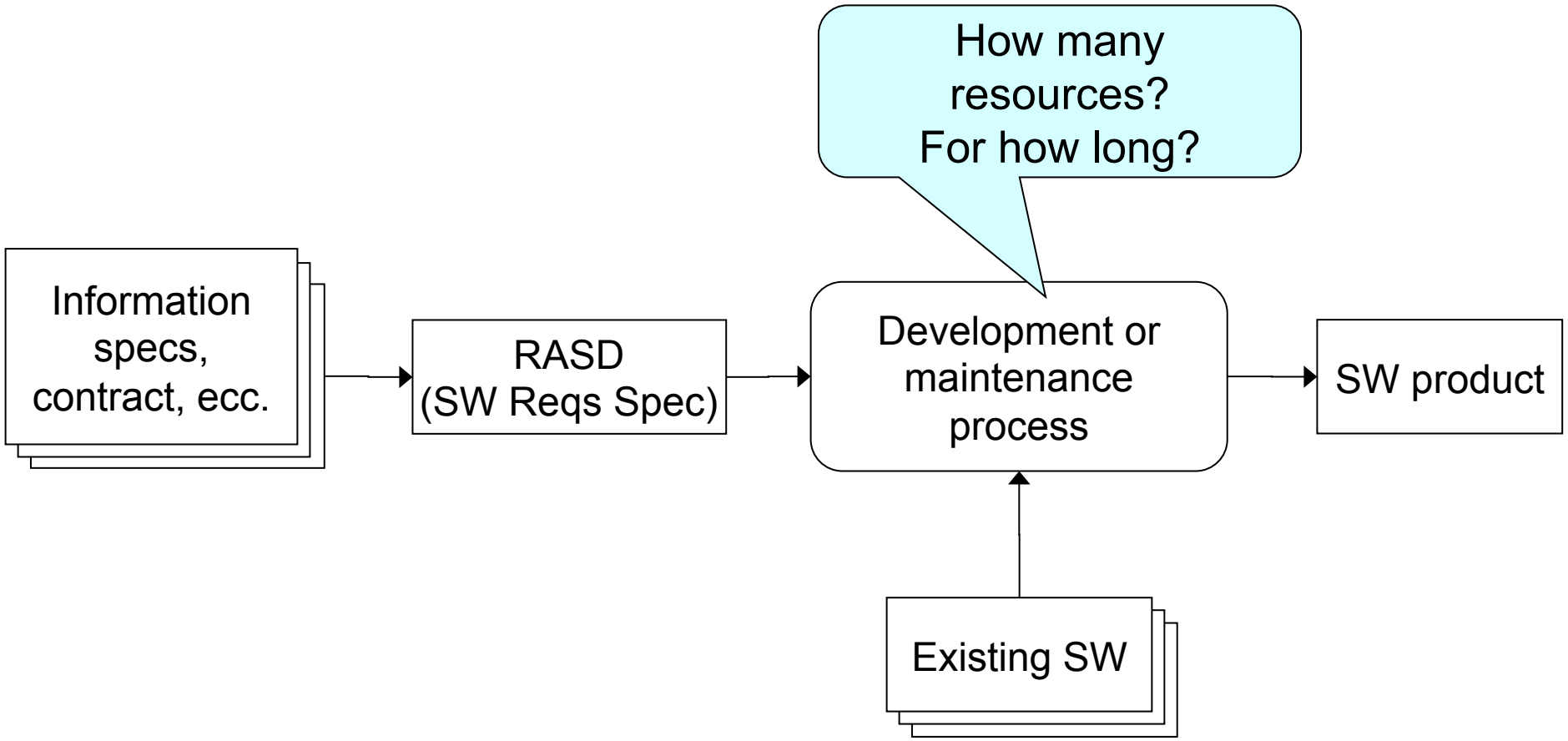
Software cost: costing and pricing (1)

- Estimates are made to discover the cost, to the developer, of producing a software system.
- There is a non-trivial relationship between the development cost and the price charged to the customer.
- Broader **organisational, social, economic, political and business** considerations influence the price charged.

Software cost: costing and pricing (2)
[I. Sommerville, “Software Engineering, 7th Ed.” Ch. 2]

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 can be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to ga in a c ontract. It is better to make a sm aller than normal profit or break even than to go out of business.

The estimation problem



Estimation techniques

- Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:
  - *Experience-based techniques* The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.
  - *Algorithmic cost modeling* In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

Experience-based approaches

- Experience-based techniques rely on judgments based on experience of past projects. Steps
  - Identify the deliverables to be produced in the new project (both documents and software)
  - Document these in a spreadsheet
  - Estimate them individually
  - Compute the total effort required.
- It usually helps to get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate.

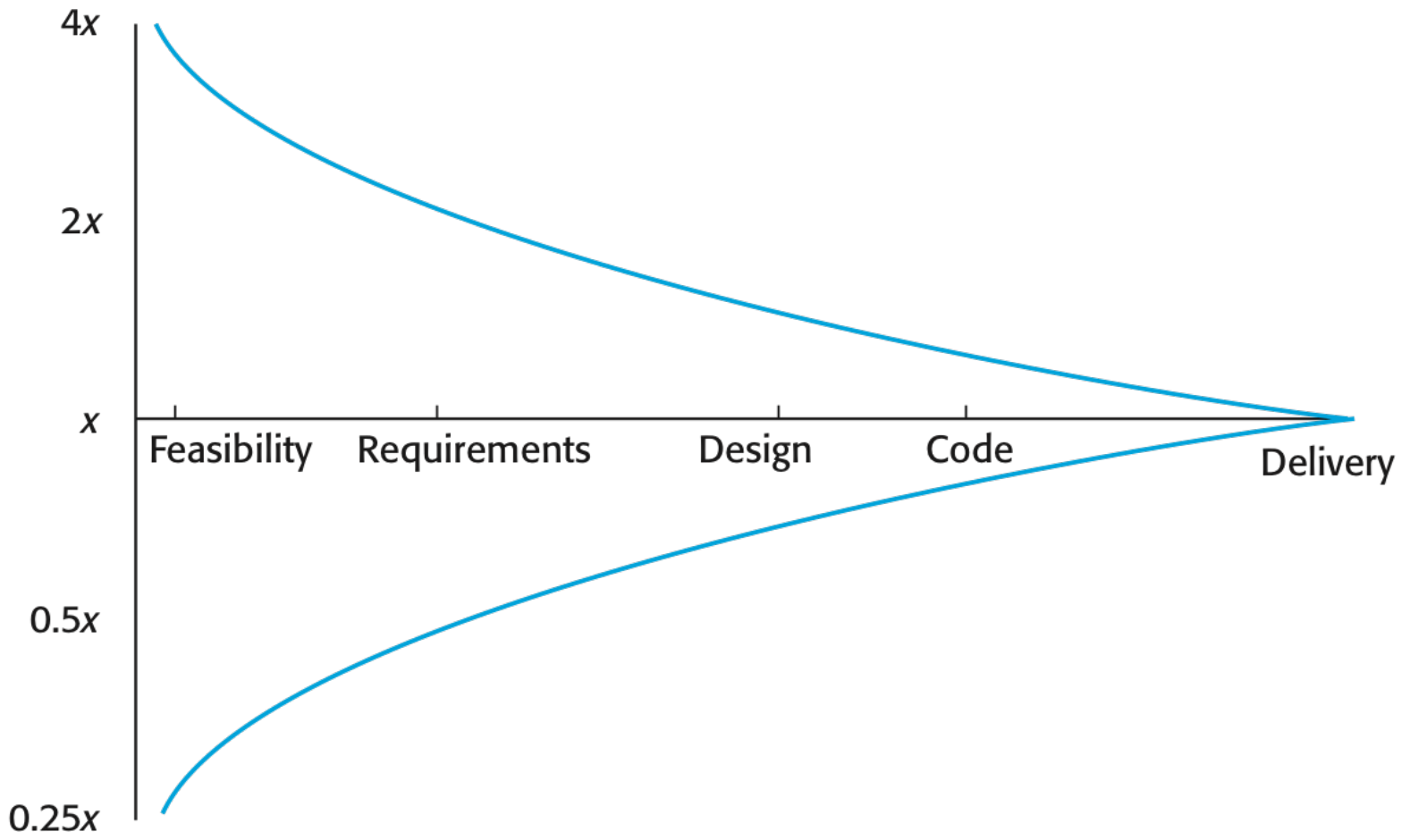
Algorithmic cost modelling

- Cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers:
  - Effort = A x $Size^B$ x M, where
  - A is an organisation-dependent constant,
  - B reflects the disproportionate effort for large projects and
  - M is a multiplier reflecting product, process and people attributes.
- The most commonly used product attribute for cost estimation is code size.
- Most models are similar but they use different values for A, B and M.

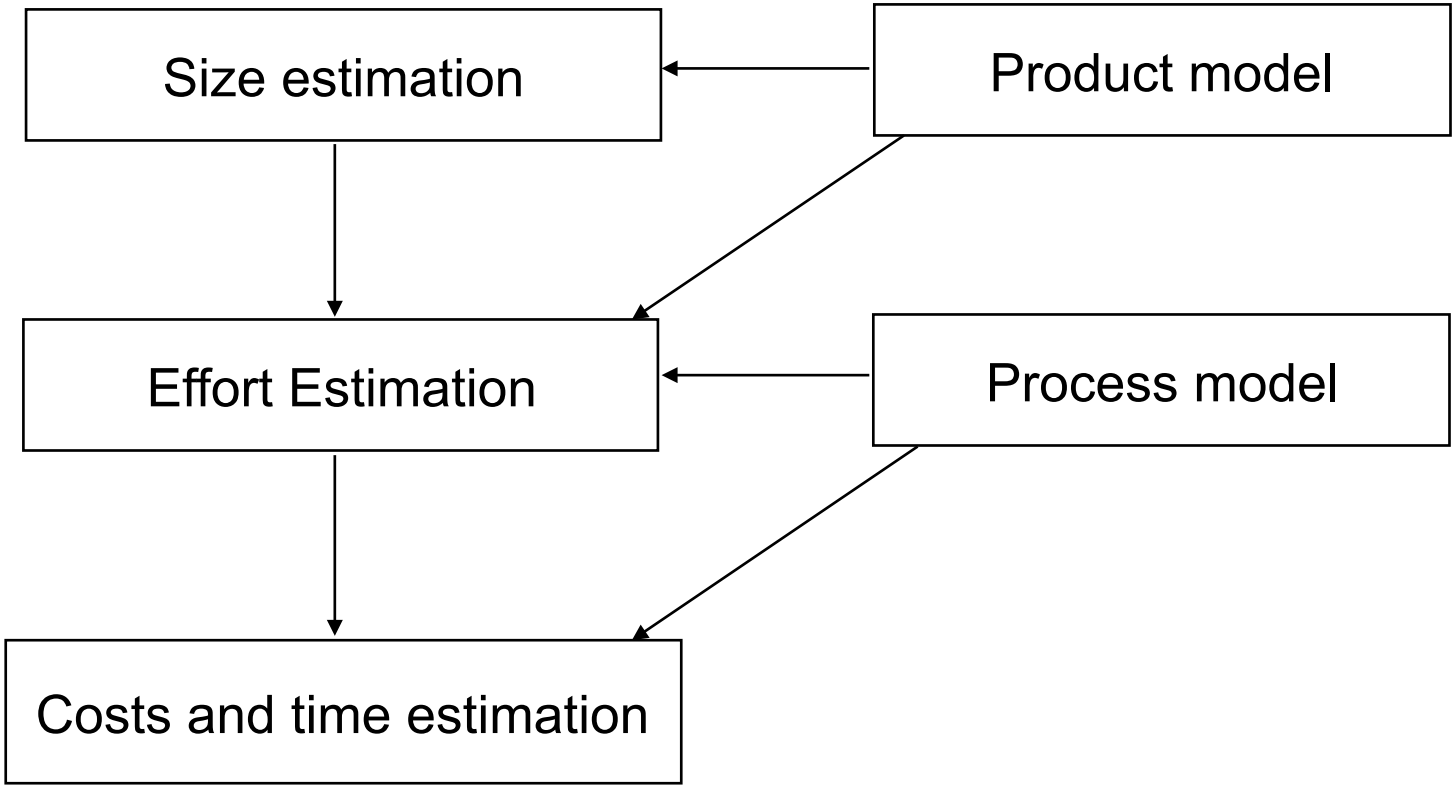
Estimation accuracy

- The 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 the team.
- As the development process progresses then the size estimate becomes more accurate.
- The estimates of the factors contributing to B and M are subjective and vary according to the judgment of the estimator.

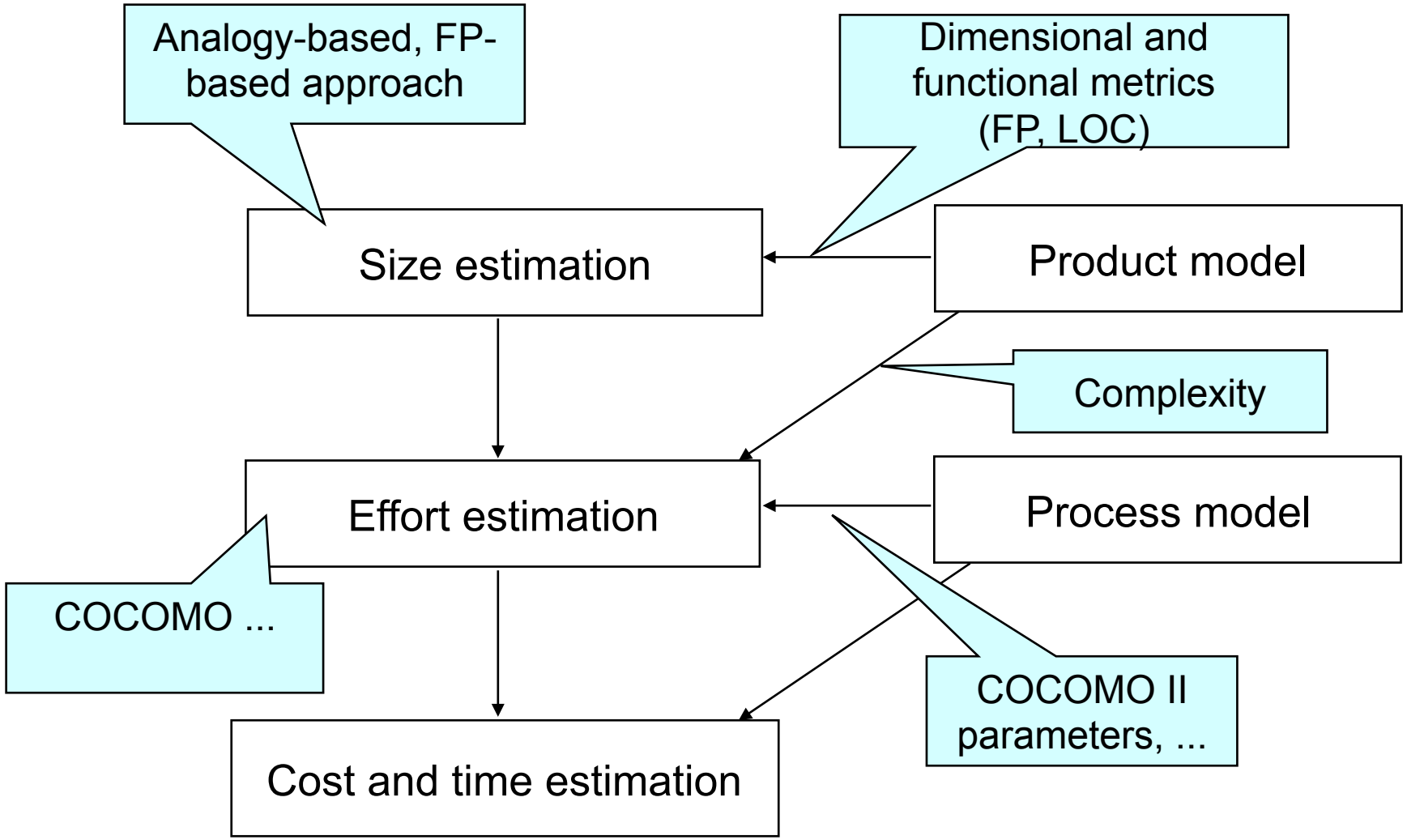
Estimate uncertainty



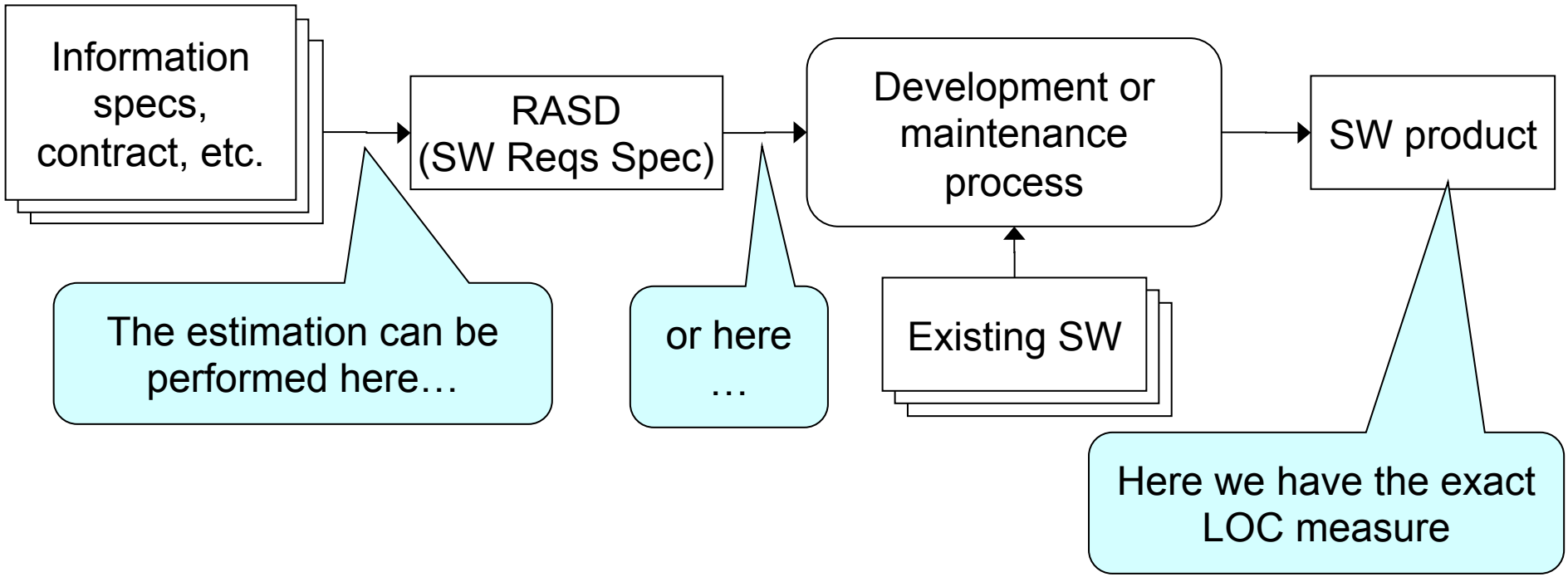
A possible estimation procedure



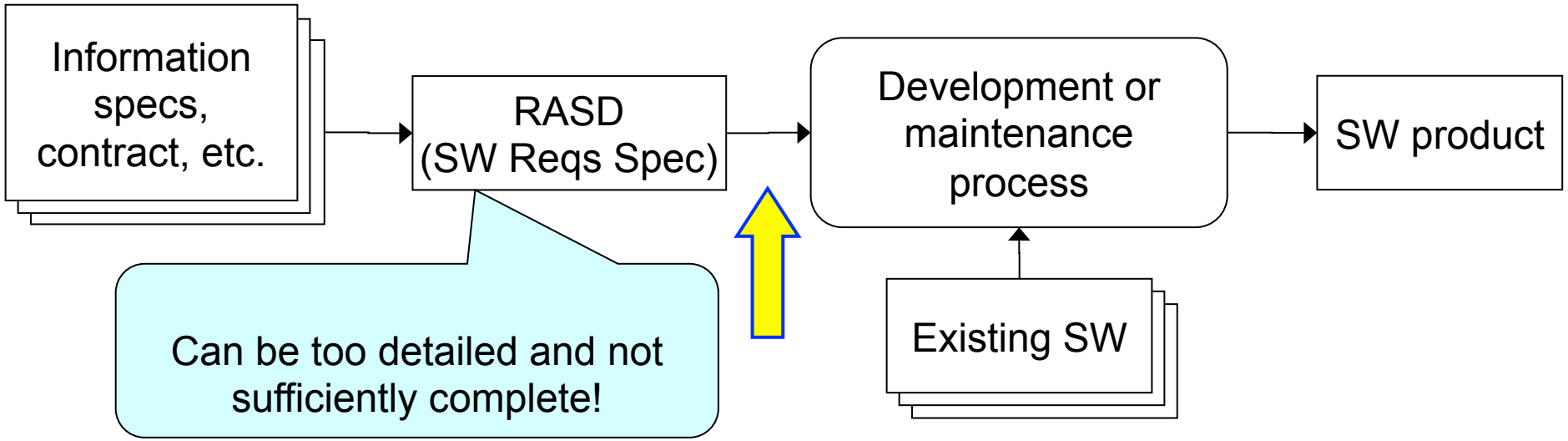
Techniques to support the estimation procedure



Size Estimation



RASD-based size estimation



- Options:
  - Calculate the number of Function Point
    - This can be used directly as a measure of the software dimension
  - Estimate the LOC
    - By converting the FP into LOCs
    - Directly
  - Any estimation has to be completed by a complexity analysis

Function points

- The Function Points approach has been defined in 1975 by Allan Albrecht (IBM)
- Basic assumption
  - the dimension of software can be characterized based on the functionalities that it has to offer

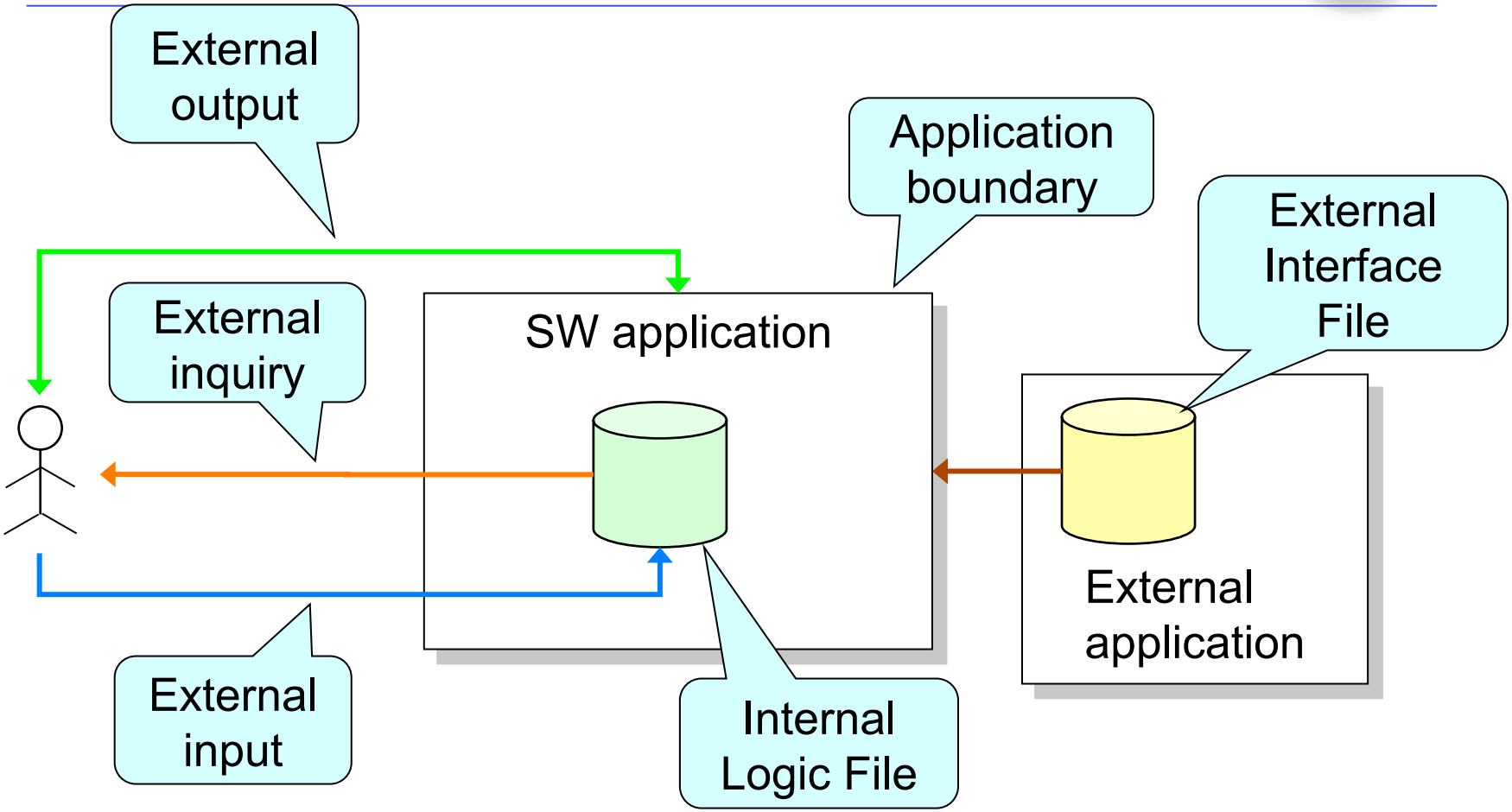
The original definition

“a technique to assess the effort needed to design and develop custom software applications”

What are Function Points (FP)?

- Based on a combination of program characteristics
  - Data structures;
  - Inputs and outputs;
  - Inquiries;
  - External interfaces;
- A weight is associated with each of these FP counts; the total is computed by multiplying each “raw” count by the weight and summing all partial values:
- UFP = $\sum$ (number of elements of a given type * weight)

Function Types (1)



Function Types (2)

- The Albrecht's method identifies and count the number of function types
- They, all together, constitute the "external representation" of an application, that is, its functionality
- Function types
  - Internal Logical File (ILF): homogeneous set of data used and managed by the application
  - External Interface File (EIF): homogeneous set of data used by the application but generated and maintained by other applications

Function Types (3)

- External Input
  - Elementary operation to elaborate data coming from the external environment
- External Output
  - Elementary operation that generates data for the external environment
    - It usually includes the elaboration of data from logic files
- External Inquiry
  - Elementary operation that involves input and output
    - Without significant elaboration of data from logic files

How was the approach defined?

- Albrecht has considered 24 applications
  - 18 COBOL, 4 PL/1, 2 DMS
  - …ranging from 3000 to 318000 LOC,
  - …from 500 to 105200 person hours
- Based on these he has defined the number of FP as the sum of Function Types weighted in order to correlate them to the development effort

Weighting function points

Complexity is evaluated based on the characteristics of the application

Function Types	Weight		
	Simple	Medium	Complex
N. Inputs	3	4	6
N. Outputs	4	5	7
N. Inquiry	3	4	6
N. ILF	7	10	15
N. EIF	5	7	10

- We obtain the Unadjusted Function Points (UFP)

Example

- Calculate FPs for an application managing a mobile telephony service. The application:
  - Manages data about both users and the different types of rates. The system will store the user data (phone number, personal data, the type of rate, any applying promotion). Each rate type is characterized by: call cost, cost of text messages, cost of surfing the Internet. Promotions change some of the cost items only for a well-defined period of time.
  - Allows users to access their information, change rates, and activate promotions.
  - Manages user billing based on information retrieved from the network management system. This last system sends to the service management system data concerning the voice calls (call duration, and the user location in case of roaming), SMSs (the information that messages have been sent, and the corresponding user location in case of roaming), and Internet surfing (connection duration, amount of downloaded data, user location) of each user.

ILFs

- The application stores the information about
  - *users*,
  - *rates*, and
  - *promotions*.
  - As the application also manages billing information, it will have to produce *invoices*.
- Each of these entities has a simple structure as it is composed of a small number of fields. Thus, we can decide to adopt for all the four the simple weight.
- Thus, 4 x 7 = 28 FPs concerning ILFs.

ELFs

- The application manages the interaction with the network management system for acquiring information about
  - *calls*,
  - *SMSs*, and
  - *Internet usage*.
- Three entities with a simple structure. Thus, we decide to adopt a simple weight for the three of them.
- We get 3 x 5 = 15 FPs.

External inputs (1)

- The application interacts with the customer:
  - Login/logout: these are simple operations, so we can adopt the simple weight for them. 2 x 3 = 6 FPs
  - Select a rate: this operation involves two entities, the rate and the user. It can still be considered simple, so, again we adopt the simple weight. 1 x 3 = 3 FPs
  - Select/eliminate a promotion: These two operations involve three entities, the user, the rate and the promotion. As such, we can considered them of medium complexity. 2 x 4 = 8 FPs

External inputs (2)

- The application also interacts with the company operators to allow them to:
  - Insert/delete information about a new rate or promotion.
  - Insert/delete information about a new user.
- Both the above operations can be consider of average complexity. 4 x 4 = 16 FPs
- In summary, we have FPs = 6+3+8+16 = 33

External inquiries

- The application allows customers to request information about
  - their profiles
  - the list of rates, promotions, and invoices that have been defined for them.
- The application also allows the operator to visualize the information of all customers.
- In summary, we have 5 different external inquiries that we can consider of medium complexity. Thus, 5 x 4 = 20 FPs.

External outputs

- The application allows the creation of invoices.
- This is a quite complex operation as it needs to collect information from IFLs and EFLs.
- Thus, we can apply the weight for the complex cases: FP = 1 x 7.

Total number of FPs

- ILFs: 28
- ELFs: 15
- External Inputs: 33
- External Inquiries: 20
- External Outputs: 7
- Total: 103

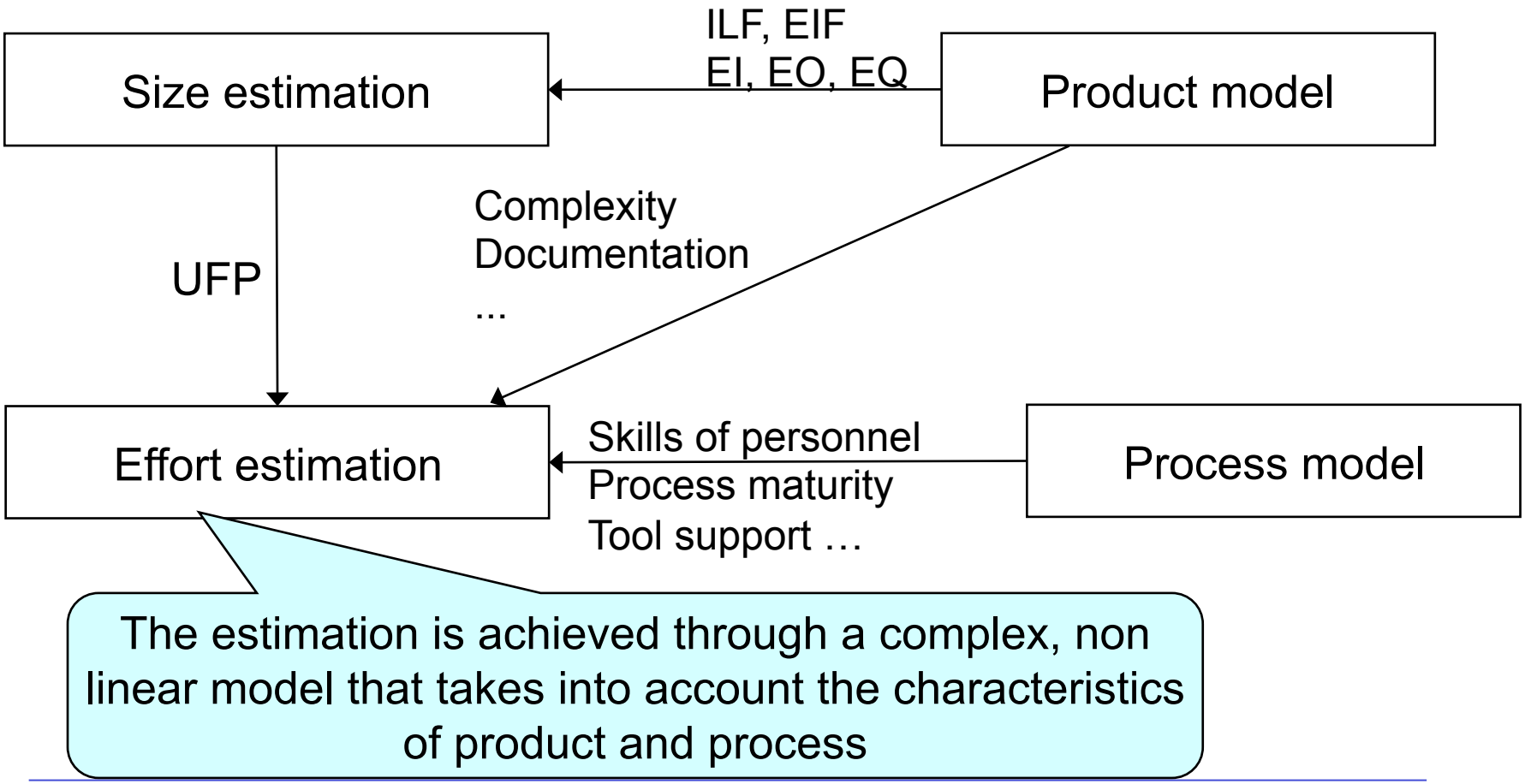
Final remarks

- The function point count is modified by complexity of the project
- FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language
  - LOC=AVC*number of function points;
    AVC is a language-dependent factor varying from 200-300 for assembly language to 2-40 for a 4GL;
  - See here for a possible mapping http://www.qsm.com/resources/function-point-languages-table
- FPs are very subjective. They depend on the estimator

The International Function Point Users Group

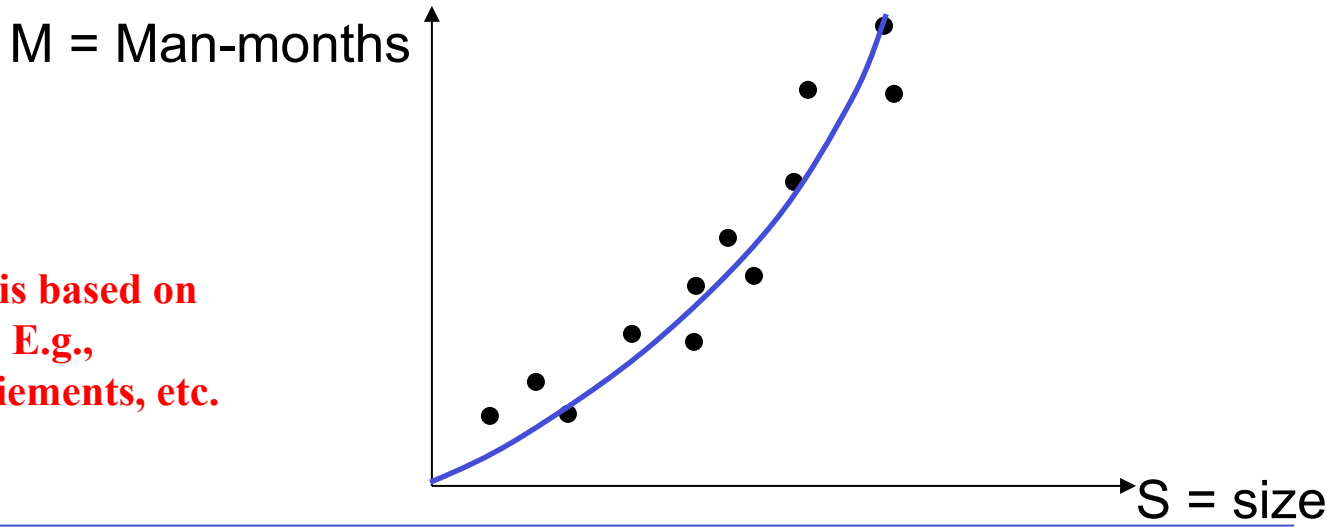
- Mission: takes care of the evolution of FPs
- Produces a manual for supporting the adoption of the FP approach
- Defines various versions of FP to apply them outside the initial context
- Offers examples useful as a reference
- http://www.ifpug.org/

COCOMO 81 (COnstructive COst Model v.1981): positioning



COCOMO 81: How was it defined?

- Barry Boehm has adopted a statistical approach to calculate a basic formula*;
- Given the size and effort of a number of homogeneous projects (same complexity) Boehm has been looking for the best correlation;
  - The linear regression applied to the logarithms of variables has offered the best results:



***COCOMO 81 model is based on outdated assumptions! E.g., waterfall, stable requriements, etc.**

COCOMO II

- Evolution of COCOMO 81 that takes into account some of the new characteristics of software development activities
- Focused on effort estimation in two cases
  - *Post-Architecture* when we are extending an existing product/product line
    - We have detailed information on cost driver inputs, and enables more accurate cost estimates.
  - *Early Design* when we do not have clear information on the architecture of the system
    - For instance, we are exploring different architectural alternatives or incremental development strategies.
    - We do not have detailed information. Thus, the estimation will be less accurate.

The main COCOMO formula: Effort Equation

- $PM = A \times Size^{E} \times \prod_{1<=i<=n} EM_i$
  - Where
    - A = 2.94. This approximates a productivity constant in PM/ KSLOC (Person-Months/Kilo-Source Lines of Code)
    - Size is the estimated size of the project in KSLOC. It can be deducted from UFP
    - EM is for Effort Multiplier. The method offers an approach to derive them from *Cost Drivers*
    - E is an aggregation of five *Scale Factors*

[Coming up next… computing Scale and Cost Drivers!]

Scale factors

- *Precedentedness*: High if a product is similar to several previously developed projects
- *Development Flexibility*: High if there are no specific constraints to conform to pre-established requirements and external interface specs
- *Architecture / Risk Resolution*: High if we have a good risk management plan, clear definition of budget and schedule, focus on architectural definition
- *Team Cohesion*: High if all stakeholders are able to work in a team and share the same vision and commitment.
- *Process Maturity*: Refers to a well known method for assessing the maturity of a software organization, CMM, now evolved into CMMI (see additional material)

Scale factors

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

- $E = B + 0.01 \times \sum_{1<=j<=5} SF_j$, where $B = 0.91$

Scale factors

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	thoroughly familiar
SF_j:	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
SF_j:	5.07	4.05	3.04	2.03	1.01	0.00
RESL	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
SF_j:	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
SF_j:	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower	SW-CMM Level 1 Upper	SW-CMM Level 2	SW-CMM Level 3	SW-CMM Level 4	SW-CMM Level 5
SF_j:	7.80	6.24	4.68	3.12	1.56	0.00

- $E = B + 0.01 \times \sum_{1<=j<=5} SF_j$, where $B = 0.91$

Empirically-proven quantities!

Cost Drivers for post-architecture

- The selection of used cost drivers depend on whether we are in the case of post-architecture or early design.
- For post-architecture 16 different factors grouped in four categories
- **Product factors**
  - Required Software Reliability (RELY)
  - Data Base Size (DATA)
  - Product Complexity (CPLX)
  - Developed for Reusability (RUSE)
  - Documentation Match to Life-Cycle Needs (DOCU)
- **Platform factors**
  - Execution Time Constraint (TIME)
  - Main Storage Constraint (STOR)
  - Platform Volatility (PVOL)

Cost Drivers for post-architecture

- **Personnel Factors**
  - Analyst Capability (ACAP)
  - Programmer Capability (PCAP)
  - Personnel Continuity (PCON)
  - Applications Experience (APEX)
  - Platform Experience (PLEX)
  - Language and Tool Experience (LTEX)
- **Project Factors**
  - Use of Software Tools (TOOL)
  - Multisite Development (SITE)
- **General Factor**
  - Required Development Schedule (SCED)

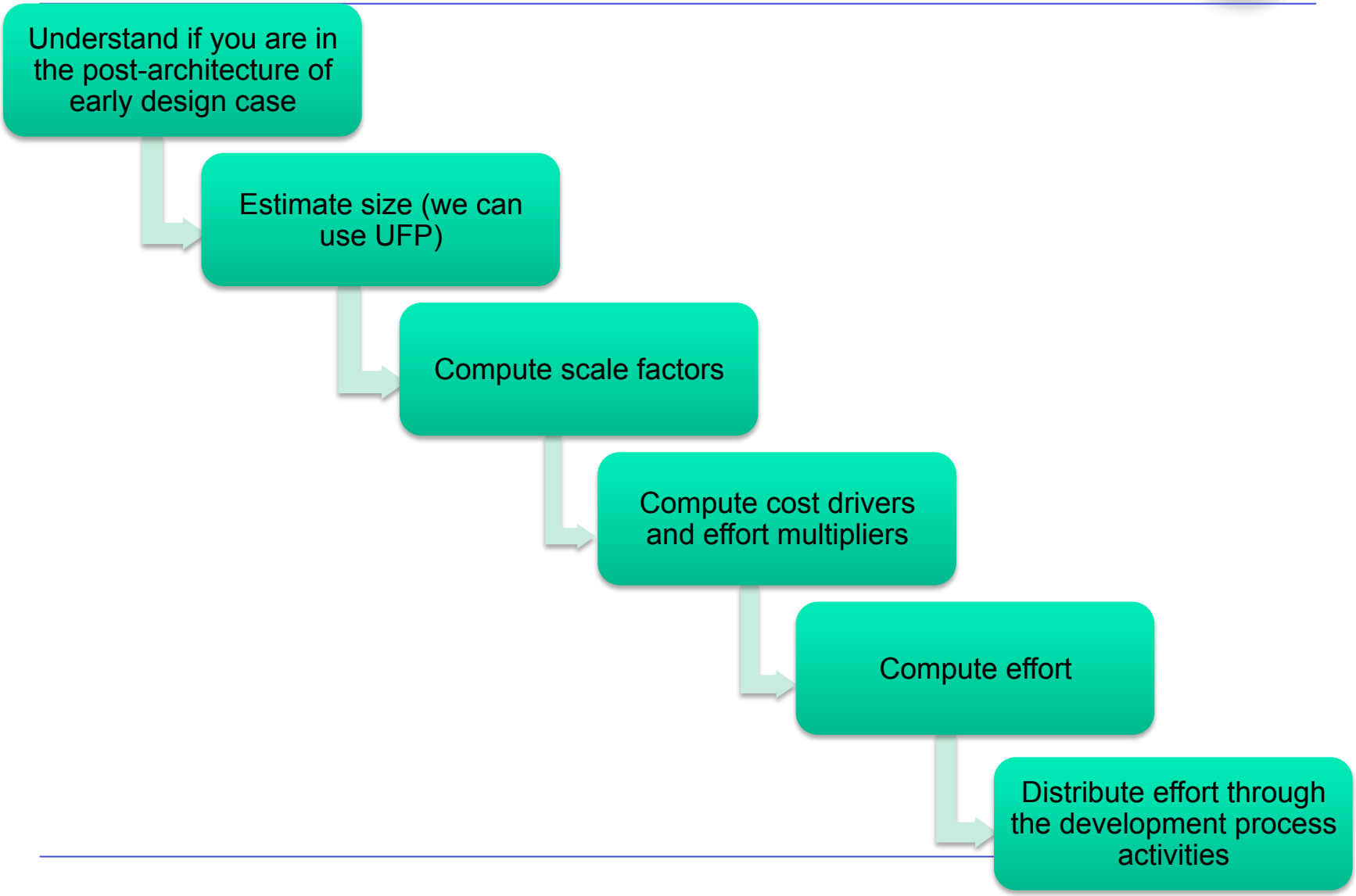
Cost Drivers for early-design

Early Design Cost Driver	Counterpart Combined Post-Architecture Cost Drivers
PERS	ACAP, PCAP, PCON
RCPX	RELY, DATA, CPLX, DOCU
RUSE	RUSE
PDIF	TIME, STOR, PVOL
PREX	APEX, PLEX, LTEX
FCIL	TOOL, SITE
SCED	SCED

From Cost Drivers to Effort Multipliers: an example

RELY Descriptors:	slight inconven-ience	low, easily recoverable losses	moderate, easily recoverable losses	high financial loss	risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

The COCOMO application process



What do we get from COCOMO?

- Never forget the statistical nature of COCOMO.
- When we apply it and get an estimation of effort, the resulting number is based on the knowledge of the current practice that is encapsulated in the model
- In other words, if the project is similar to the ones that have been considered for the definition of the model, then the computed effort is likely to be correct
  - Sometimes we need to calibrate the model

- Any Questions?