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


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.

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 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

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

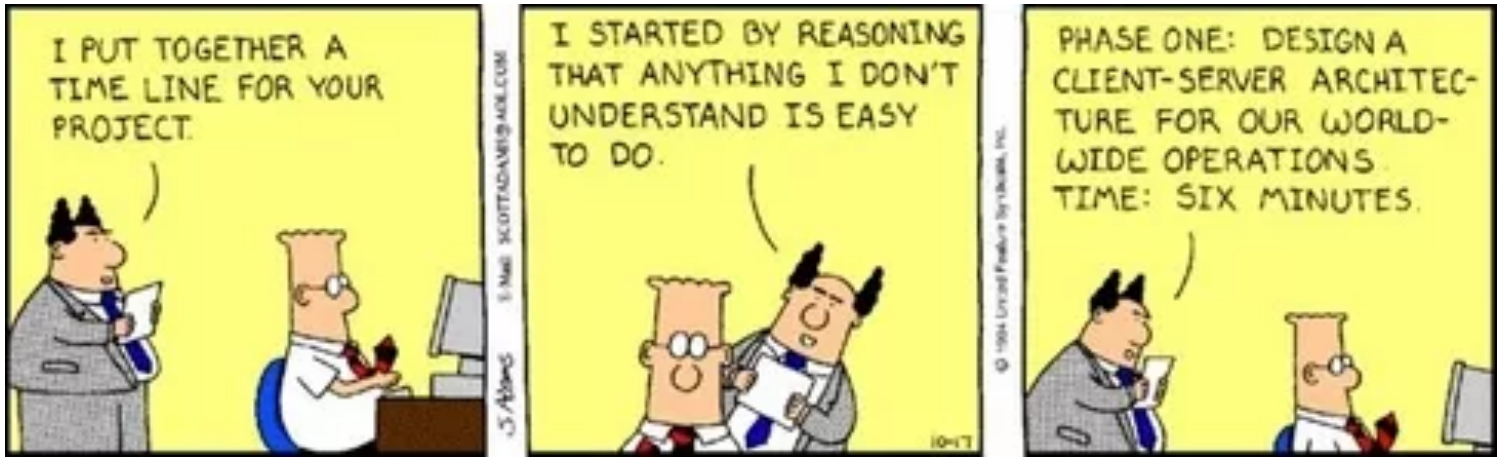
- Software Project Management needs **flexibility and adaptation**
- **Adapting means software and people**
- Management in Sw.Eng. is **as much a work of improvisation than precision**
- Sometimes (or actually most of the times) **products may well have to be ``good-enough''**

Roadmap

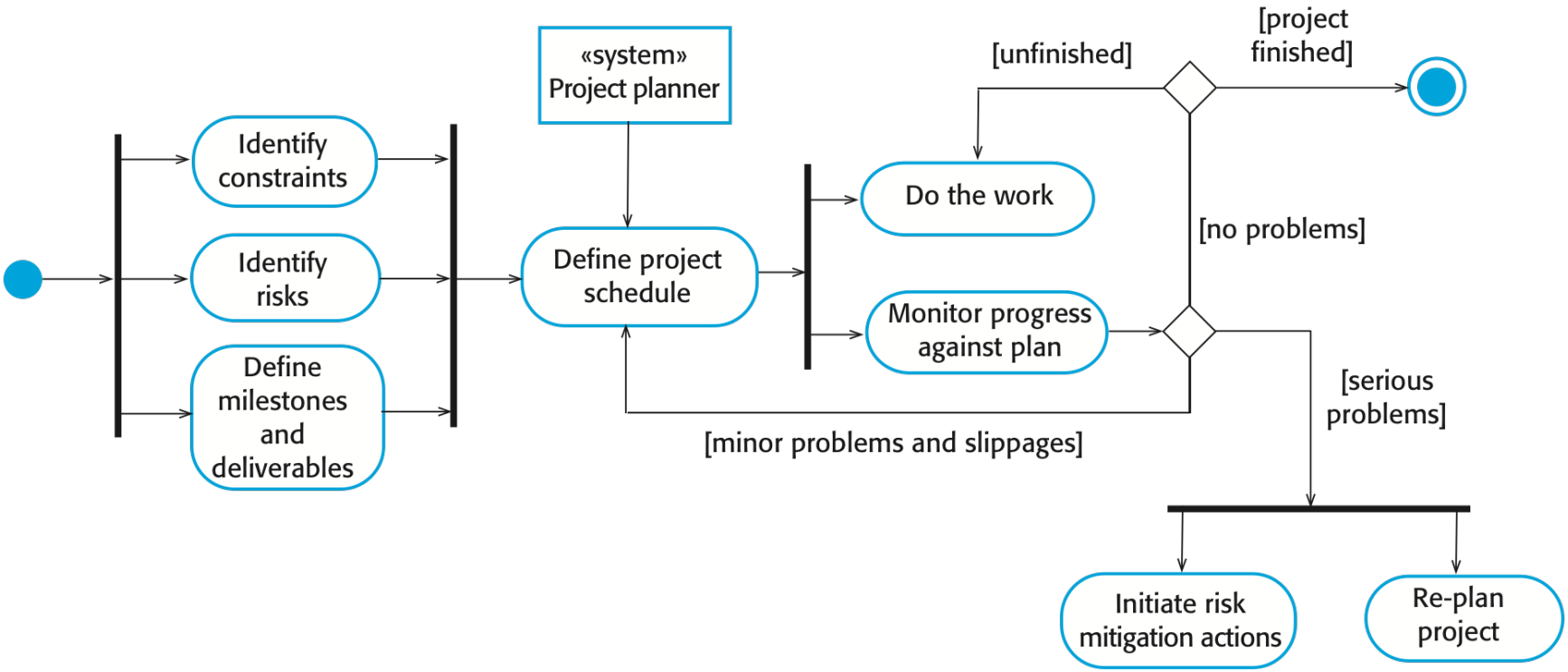
- Software Project Management
- **Project Planning**
  - Size, effort and cost estimation
- Risks and People Management
- Project and Process Improvement: CMMI

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.

The project planning process



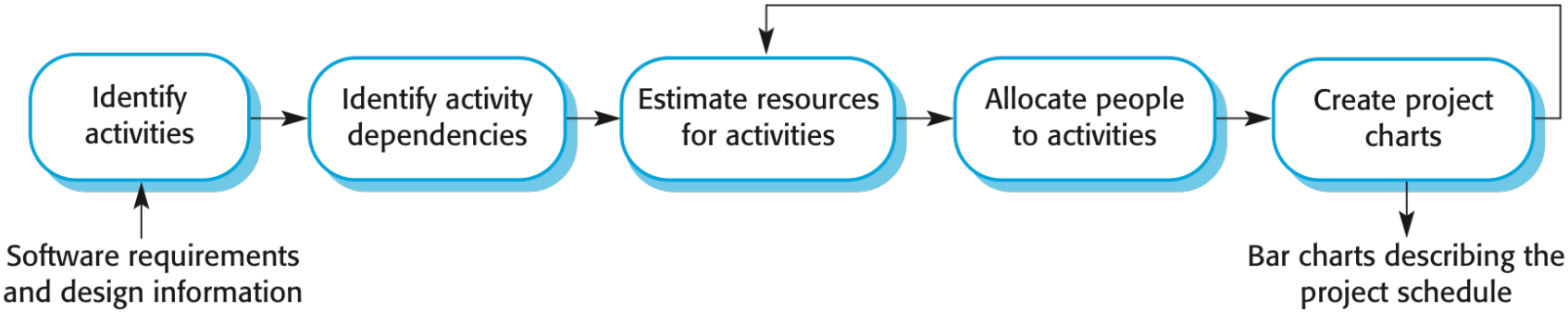
Terminology

- 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)

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.

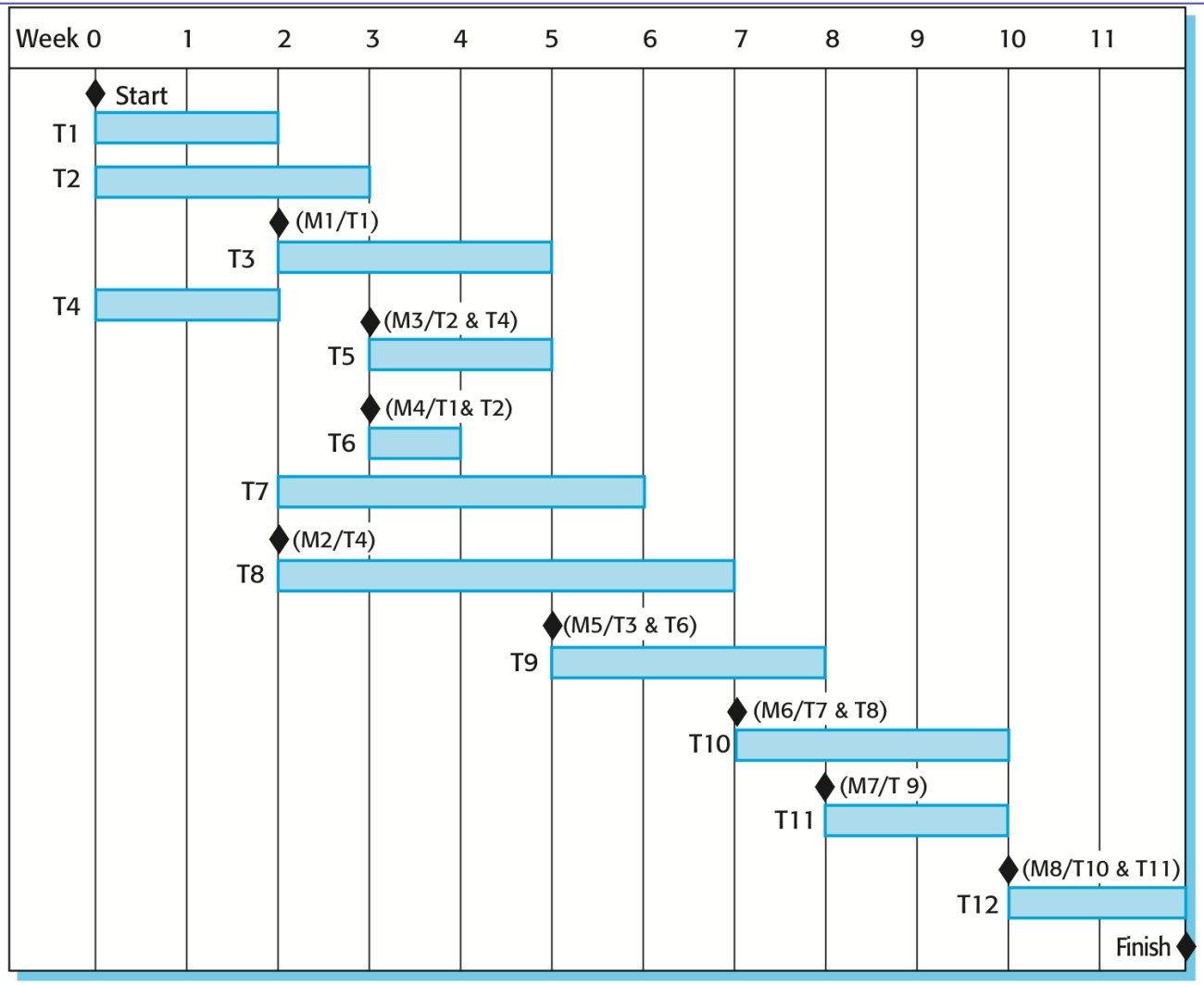
Schedule representation

- 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.

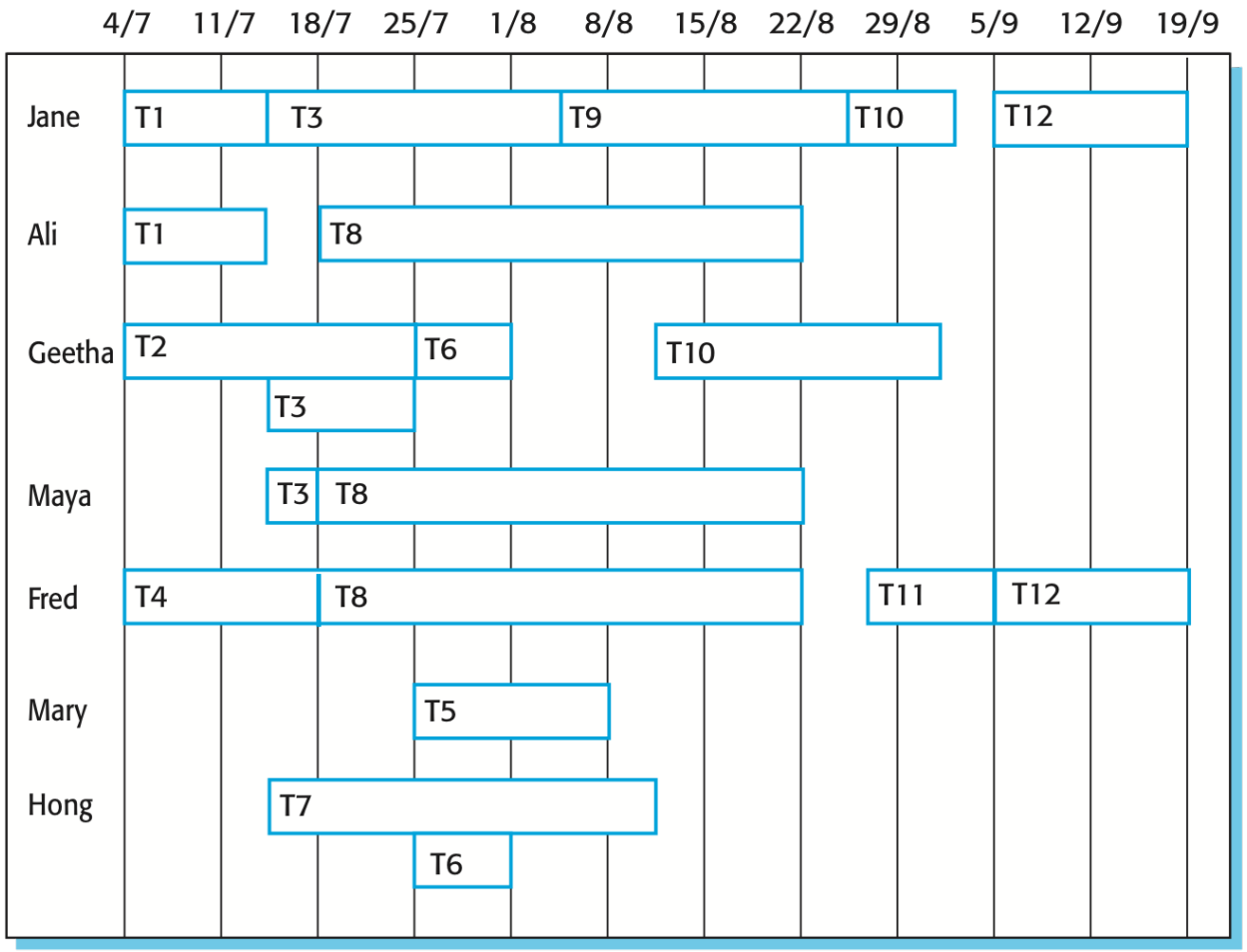
Tasks, durations, and dependencies

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

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.

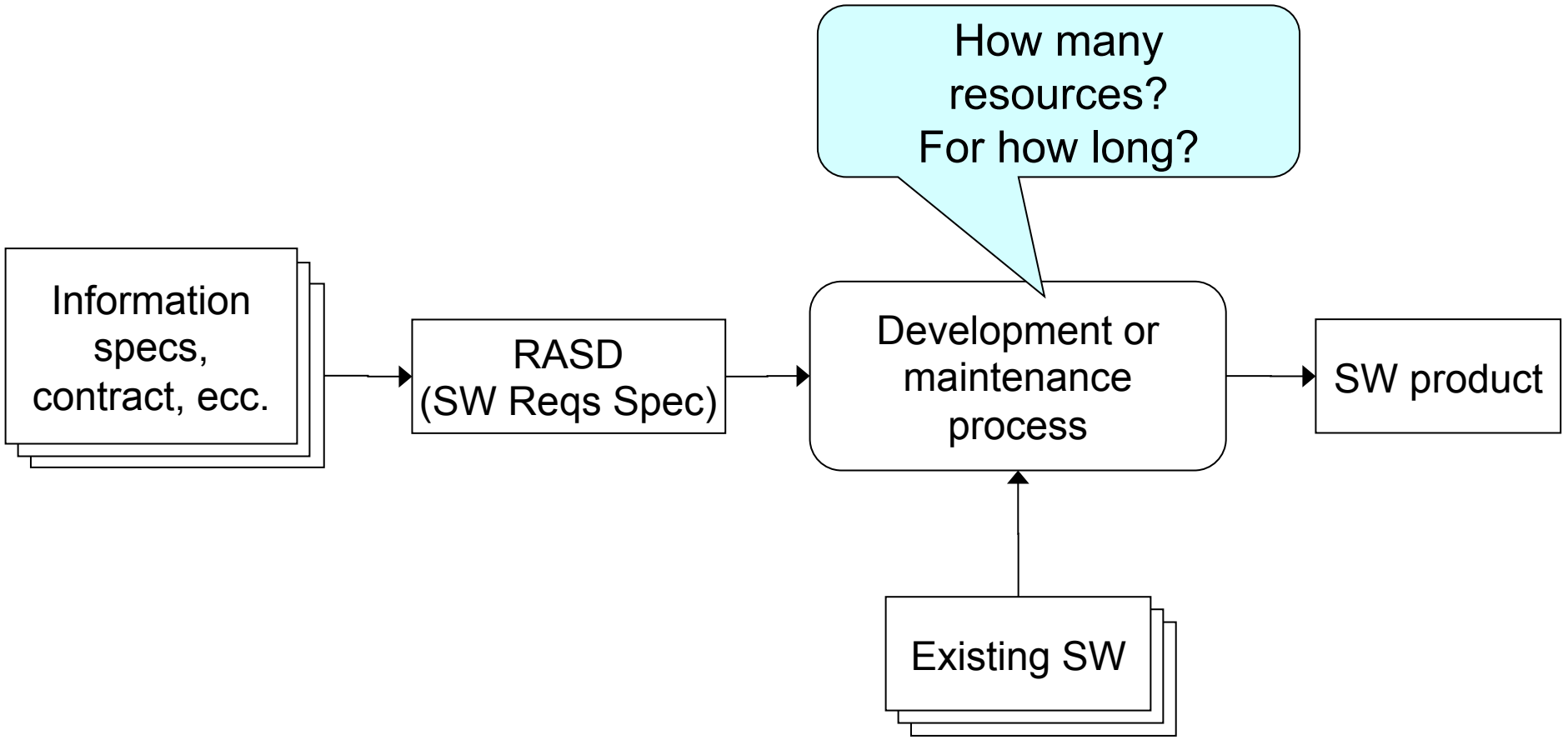
Software cost: costing and pricing (1)

- 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 gain a contract. It is better to make a smaller 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.

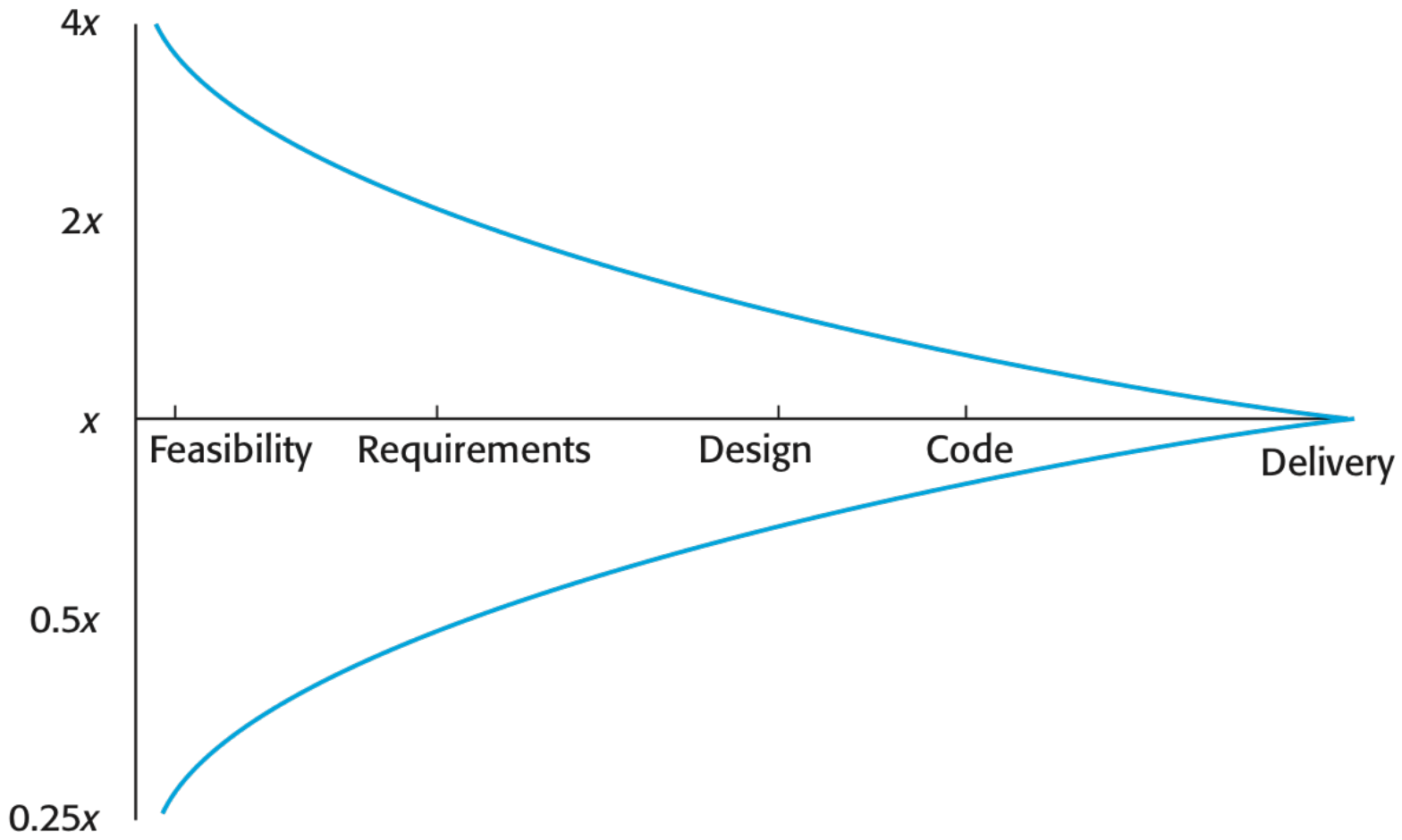
Algorithmic cost modelling

- 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.

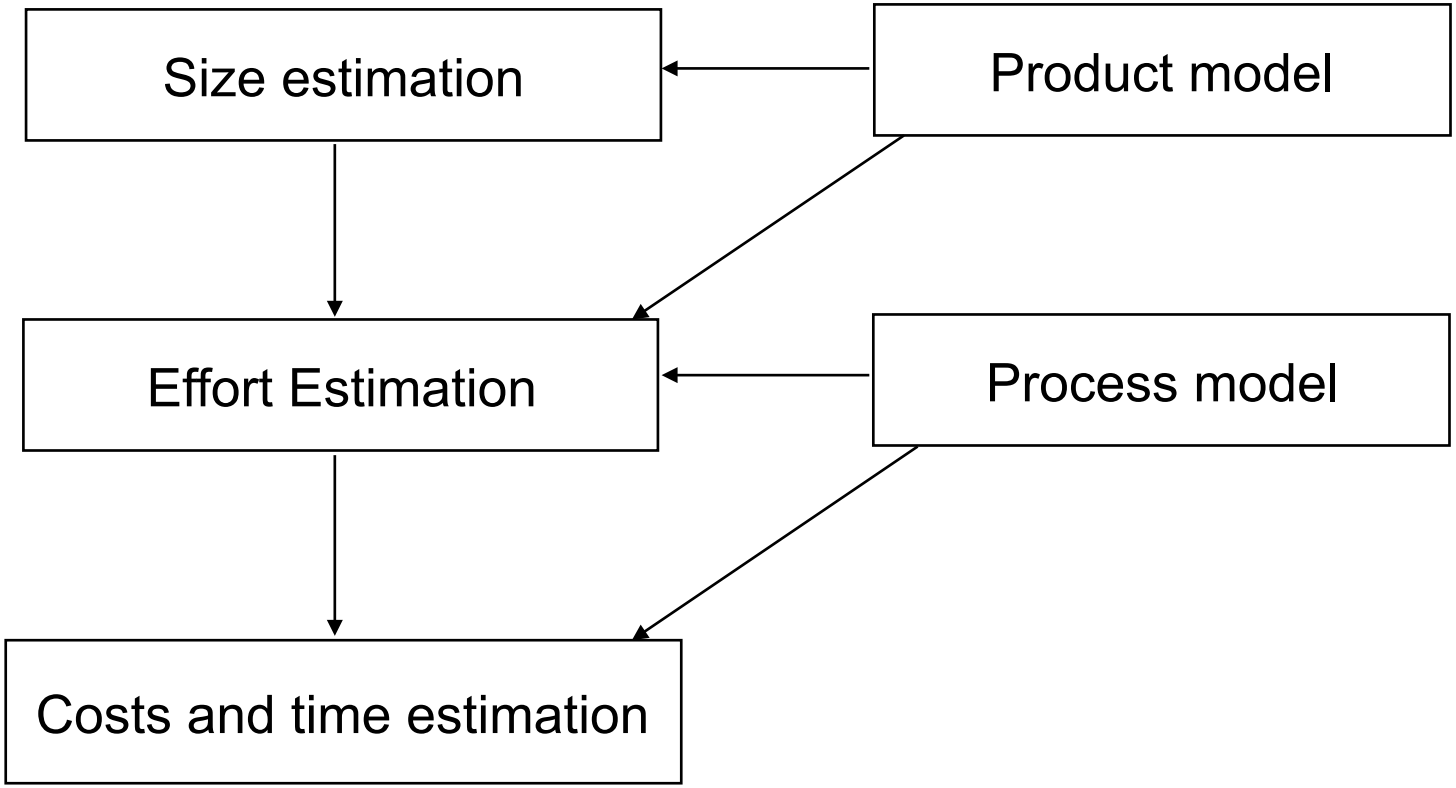
Estimation accuracy

- 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.

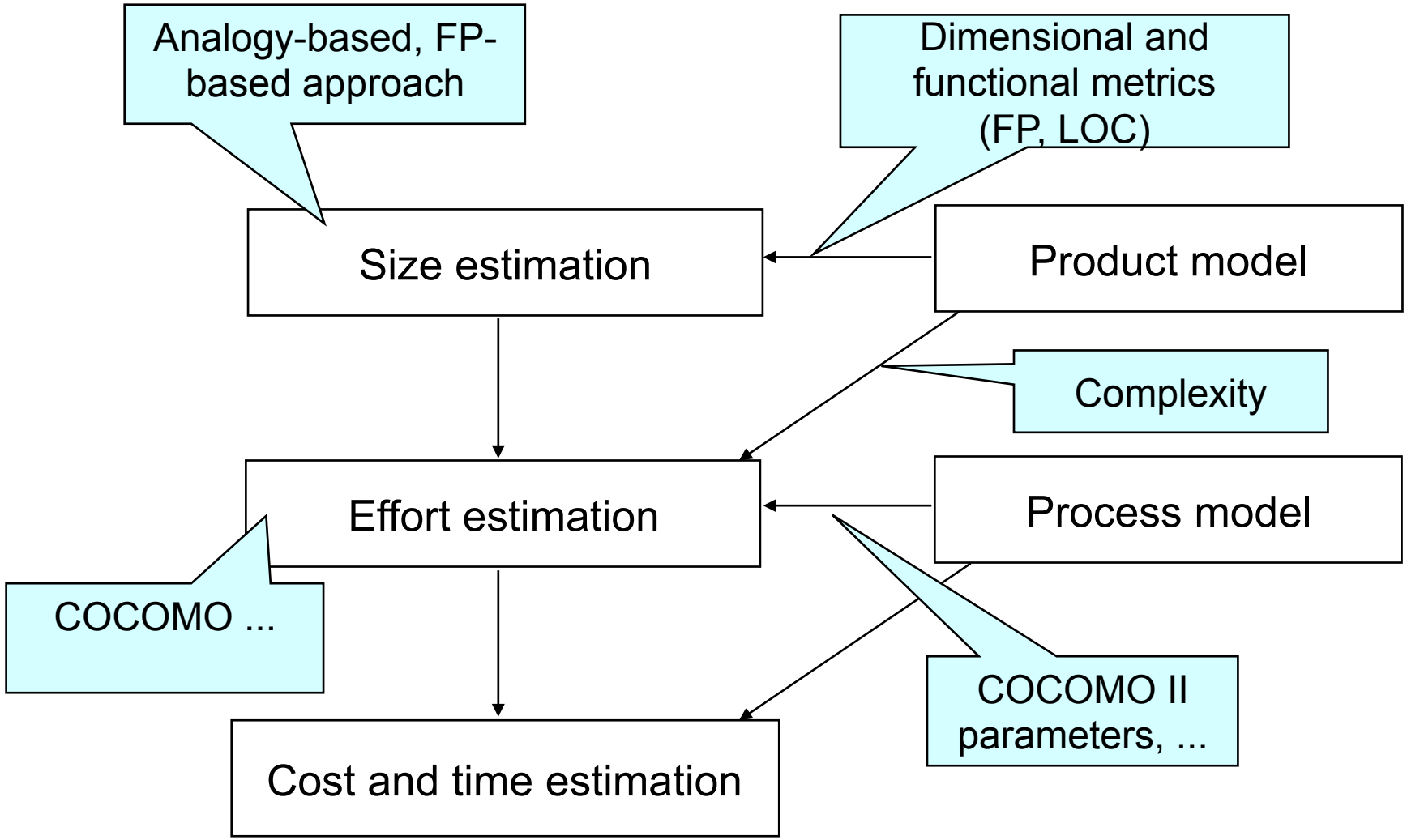
Estimate uncertainty



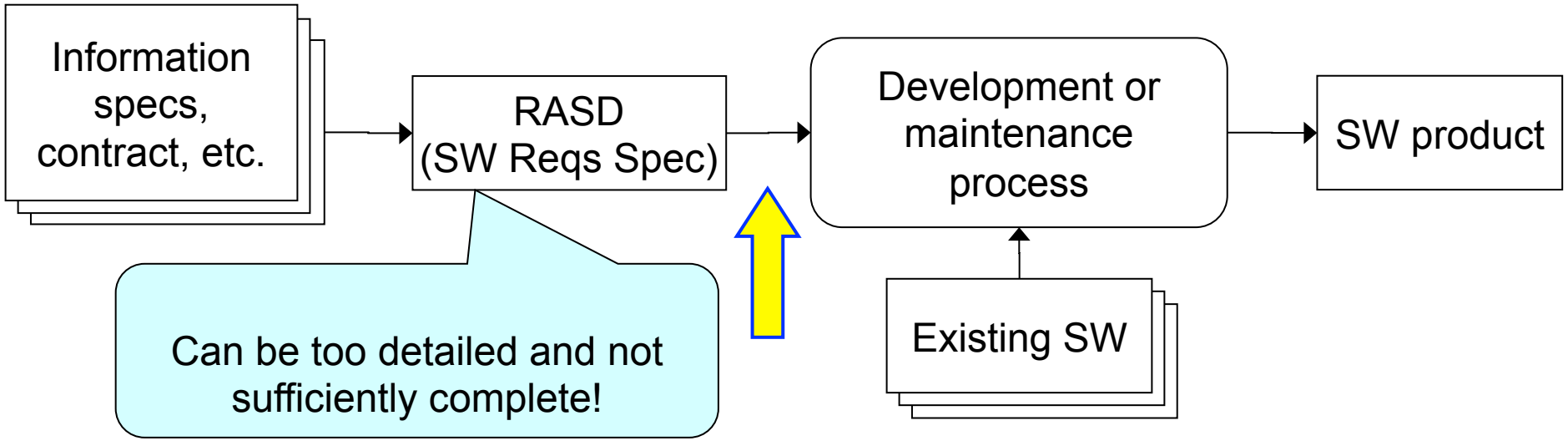
A possible estimation procedure



Techniques to support the estimation procedure



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

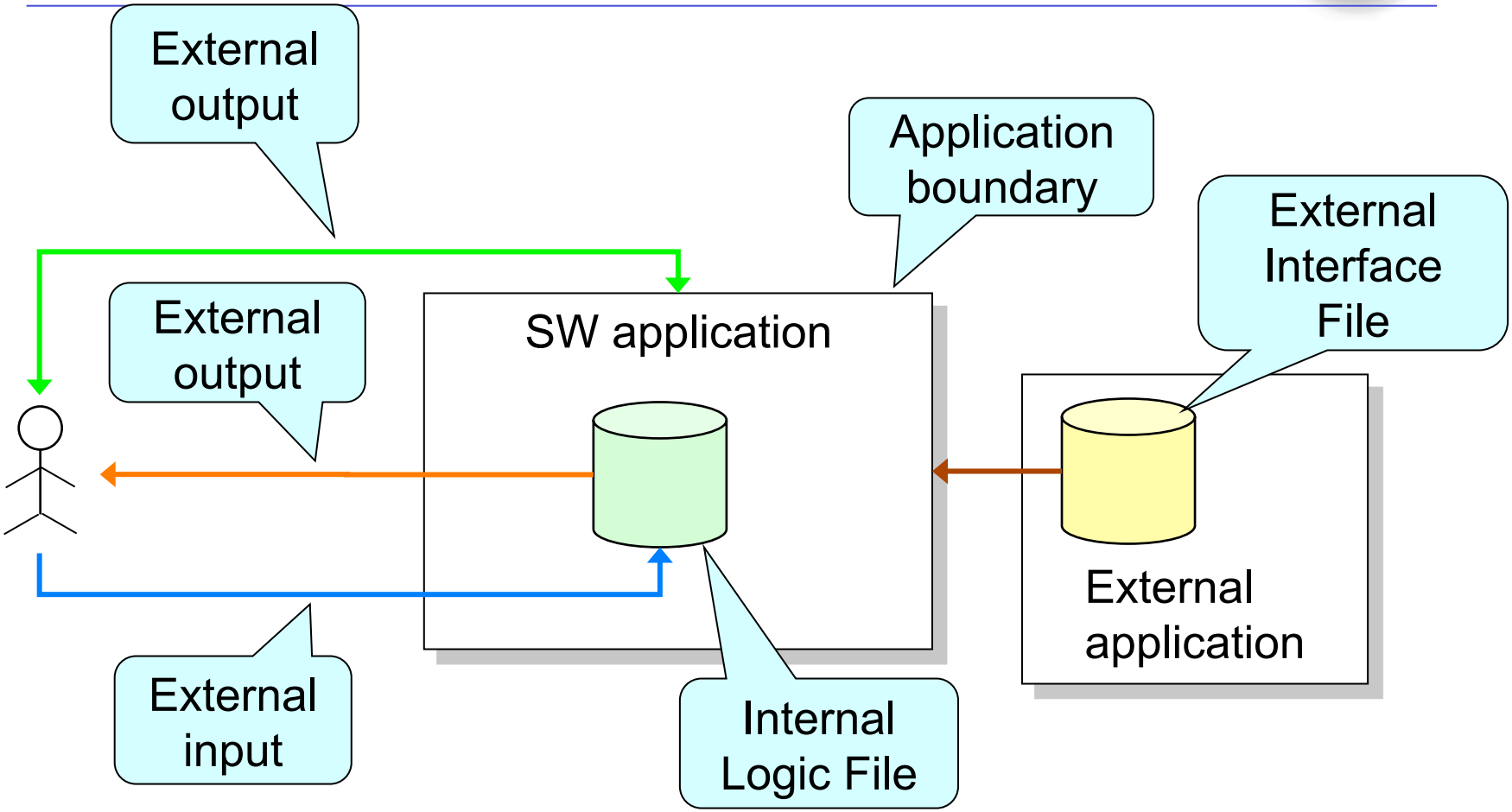
Function points

- 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

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

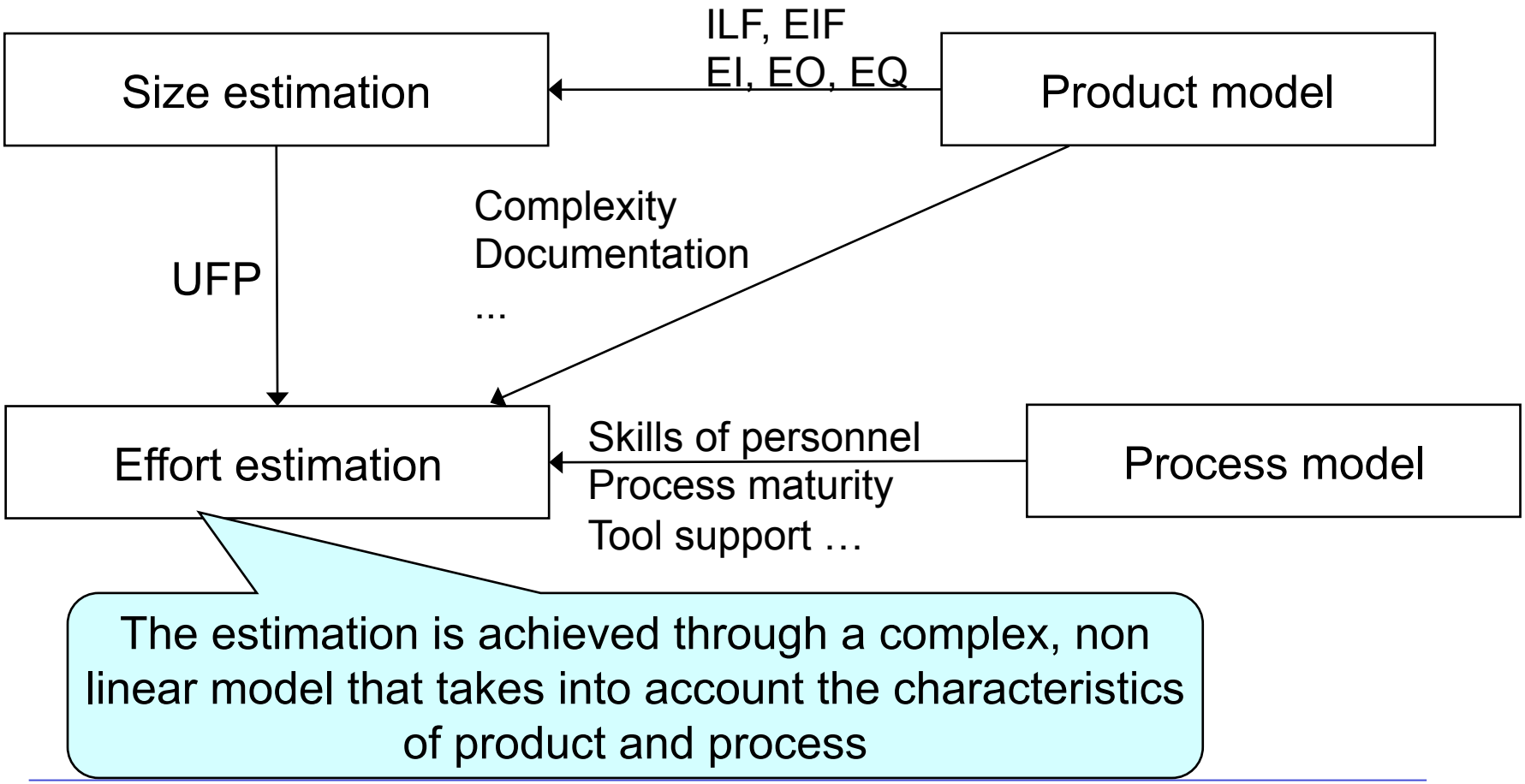
Final remarks

- 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

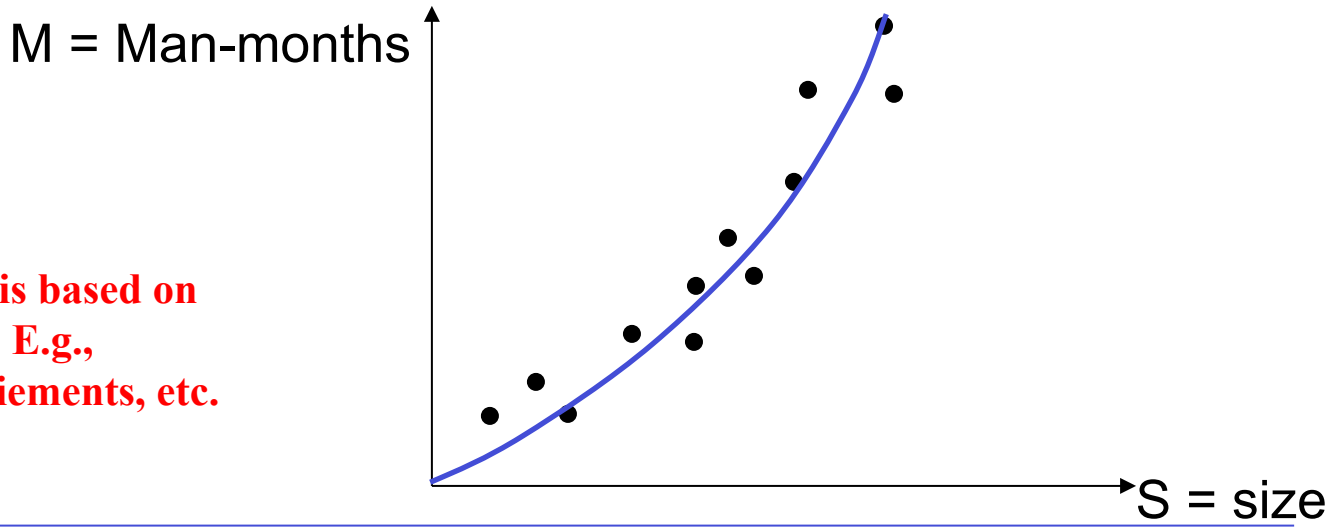
- Mission: takes care of the evolution of FPs
- Produces a manual for supporting the adoption of the FP approach
- Defines various versions of FP to apply them outside the initial context
- Offers examples useful as a reference
- http://www.ifpug.org/

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)

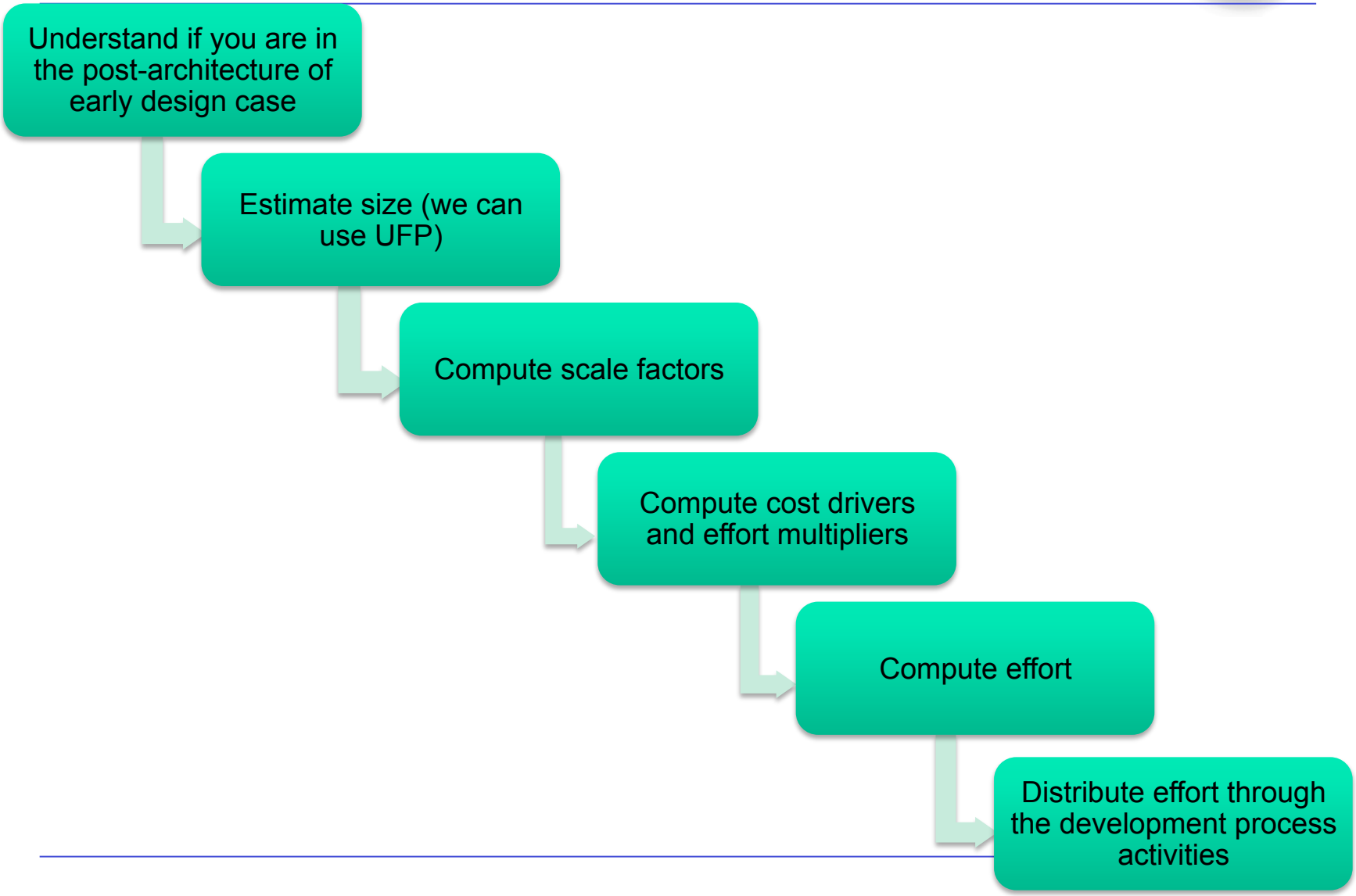
Cost Drivers for early-design

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?

Politecnico
di Milano

More on… 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

But before that… COCOMO 2: an Exercise!

- The software development project in this case study was conducted in a commercial environment with an external customer financing the software development;
- Software engineering consultants from the supplier's organization primarily compiled the project with some additional technical experts from a subcontractor;
- Domain experts and software engineers from the customer's organization also ensue;
- Development Community:
  - 26 project members;
- Project period:
  - 12 months;

Bootstrapping COCOMO II

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \text{ x } \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

Bootstrapping COCOMO II

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

$$E = B + 0.01 \sum_{j=1}^{5} SF_j$$

B = 0.91 (for COCOMO II.2000)

Step 1: Scale Factors!

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

$$A = 2.94 \text{ (for COCOMO II.2000)}$$

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

➔ **1. Precedentedness**
➔ **2. Flexibility**
➔ **. . .**

$$E = B + 0.01 \sum_{j=1}^{5} SF_j$$

$$B = 0.91 \text{ (for COCOMO II.2000)}$$

Step 1: Scale Factors - Precedentedness

Feature	Very Low	Nominal/High	Extra High
[1] Organizational understanding of product objectives	General	Considerable	Thorough
[2] Experience in working with related software systems	Moderate	Considerable	Extensive
[3] Concurrent development of associated new hardware and operational procedures	Extensive	Moderate	Some
Need for innovative data processing architectures, algorithms	Considerable	Some	Minimal

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

→ **1. Precedentedness**

→ 2. Flexibility

→ . . .

Step 1: Scale Factors – Precedentedness, Rationale

- The organization has extensive relationship with customer and related products. developers are working at the client's location;
- The system is an extension of an application framework that was developed prior to the project in order to increase productivity;
- New components are added to the framework concurrently with the application development if required. The scope is those kinds of operational procedures.

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

→ **1. Precedentedness**

→ 2. Flexibility

→ . . .

Step 1: Scale Factors - Flexibility

Feature	Very Low	Nominal/High	Extra High
[1] Need for software conformance with preestablished requirements	Full	Considerable	Basic
[2] Need for software conformance with external interface specifications	Full	Considerable	Basic
[3] Combination of inflexibilities above with premium on early completion	High	Medium	Low

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

→ 1. Precedentedness

→ **2. Flexibility**

→ . . .

Step 1: Scale Factors – Flexibility, Rationale

- Customers do not have a clear depiction of what the new software is intended to do;
- The system has indeed immense conformance constraints with external interface specifications in operational ecosystem;
- The system needs to be up and running by the time specified to the contractor's customers;

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

→ 1. Precedentedness

→ **2. Flexibility**

→ . . .

Step 1: Scale Factors – proceed until done!

- …

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

→ 1. Precedentedness

→ 2. Flexibility

→ …

Step 2: Cost Drivers!

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times Size^{E} \prod_{i=1}^{17} EM$$

A = 2.94 (for COCOMO II.2000)

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

➔ **1. Product**
➔ **2. Platform**
➔ **…**

$$E = B + 0.01 \sum_{j=1}^{5} SF_j$$

B = 0.91 (for COCOMO II.2000)

Step 2: Cost Drivers!

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times Size^{E} \prod_{i=1}^{17} EM$$

A = 2.94 (for COCOMO II.2000)

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

➔ 1. Product

➔ 2. Platform

➔ …

$$E = B + 0.01 \sum_{j=1}^{5} SF_j$$

B = 0.91 (for COCOMO II.2000)

Step 2: Cost Drivers – Platform Volatility

PVOL Descriptors		Major change every 12 mo.; Minor change every 1 mo.	Major: 6 mo.; Minor: 2 wk.	Major: 2 mo.; Minor: 1 wk.	Major: 2 wk.; Minor: 2 days	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.87	1.00	1.15	1.30	n/a

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

→ 1. Product

→ **2. Platform**

→ …

Step 2: Cost Drivers – Platform Volatility, Rationale

- The project (and organization) is specialized in working with the Oracle application framework and underlying platform. Updates in project applications are fairly rare in relation to new platform version introductions;

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

→ 1. Product

→ **2. Platform**

→ …

Step 2: Cost Drivers – iterate over cost drivers table until done!

- The project (and organization) is specialized in working with the Oracle application framework and underlying platform. Updates in project applications are fairly rare in relation to new platform version introductions;

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

→ 1. Product
→ 2. Platform
→ . . .

Summing it all up: Scale Factors

Scale Factors (SF_j)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented 6.20 (**0**)	largely unprecedented 4.96	somewhat unprecedented 3.72 (**0.5**)	generally familiar 2.48 (**0.5**)	largely familiar 1.24	thoroughly familiar 0.00 (**3**)
FLEX	rigorous 5.07 (**3**)	occasional relaxation 4.05	some relaxation 3.04 (**0**)	general conformity 2.03 (**0**)	some conformity 1.01	general goals 0.00 (**0**)
RESL[1]	little (20%) 7.07 (**2**)	some (40%) 5.65 (**2**)	often (60%) 4.24 (**0**)	generally (75%) 2.83 (**1**)	mostly (90%) 1.41 (**1**)	full (100%) 0.00 (**1**)
TEAM	very difficult interactions 5.48 (**0**)	some difficult interactions 4.38 (**0**)	basically cooperative interactions 3.29 (**1**)	largely cooperative 2.19 (**1**)	highly cooperative 1.10 (**1**)	seamless interactions 0.00 (**1**)
PMAT	(Weighted average of "Yes" answers to CMM Maturity Questionnaire) The estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	SW-CMM Level 2 **4.68**	SW-CMM Level 3 3.12	SW-CMM Level 4 1.56	SW-CMM Level 5 0.00

Following these values, the exponent E in the effort estimation equation evaluates to 1.07

Summing it all up: Cost Drivers

Cost Driver	Rating Level	Effort Multiplier
RELY	High	1.10
DATA	Very High	1.28
RUSE	Very High	1.15
DOCU	Nominal	1.00
TIME	Nominal	1.00
STOR	Low	n/a
PVOL	Low	0.87
ACAP	Extra High	n/a
PCAP	High	0.88
PCON	Very High	0.81
APEX	High	0.88
PLEX	Very High	0.85
LTEX	Very High	0.84
TOOL	Nominal	1.00
SITE	(Very Low) and (Extra High)	1.01 (0.5x1.22 + 0.5x0.80)
SCED	Nominal	1.00
CPLX	Nominal	1.00

Calibrating EM with the above value results in a factor of 0.64

Step 3: COCOMO II Effort Equation

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

$$A = 2.94 \text{ (for COCOMO II.2000)}$$

Using FPs I estimate a size of 39 (KSLOC)

Step 3: COCOMO II Effort Equation

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ Post-Architecture!

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

$$A = 2.94 \text{ (for COCOMO II.2000)}$$

Using FPs I estimate a size of 39 **(Kilo Source Lines Of Code)**

Step 3: COCOMO II Effort Equation

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

$$A = 2.94 \text{ (for COCOMO II.2000)}$$

Using FPs I estimate a size of 39 (KSLOC), Hence:

$$PM = 2.94 * (39)^{1.07} * 0.64 = \mathbf{95,6\ PM;}$$

Step 3: COCOMO II Effort Equation

- An application framework was used to obtain founds for the original idea so the architecture and general application structure were therefore available immediately at project start ➔ **Post-Architecture!**

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

Using FPs I estimate a size of 39 (KSLOC), Hence:

PM = 2.94 * $(39)^{1.07}$ * 0.64 = 95,6 PM;
In our Organization, 1PM == 152 PH;

14577 PH [ESTIMATED] for the entire project;

Here's a good question…

- How Much would you think was actually invested in this project?

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

14577 PH [ESTIMATED] for the entire project;

Here's a good question…

- How Much would you think was actually invested in this project?

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

14577 PH [ESTIMATED] for the entire project;

Actual figure was 9938 PHs… 47% gain! ☺

Here's a good question…

- How Much would you think was actually invested in this project?

$$PM = A \times \text{Size}^{E} \prod_{i=1}^{17} EM_i$$

A = 2.94 (for COCOMO II.2000)

14577 PH [ESTIMATED] for the entire project;

Actual figure was 9938 PHs… 47% gain! ☺**

** if it's too high, there's risk of ***client loss***…

Here's a good question…

- How Much would you think was actually invested in this project?

14577 PH [ESTIMATED] for the entire project;

Actual figure was 9938 PHs… 47% gain! ☺**

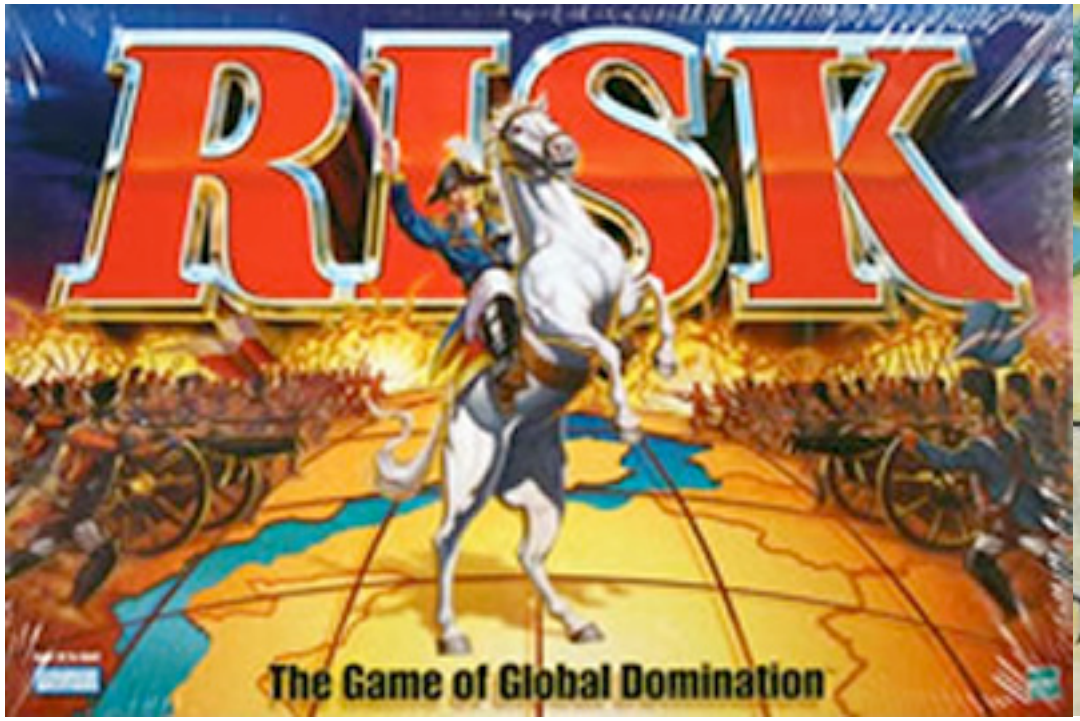
** if it's too high, there's **risk** of ***client loss…***

Thanks for your endurance…

- Break?

Coming up next… Risks and Process Improvement!

What are Software Project Risks?



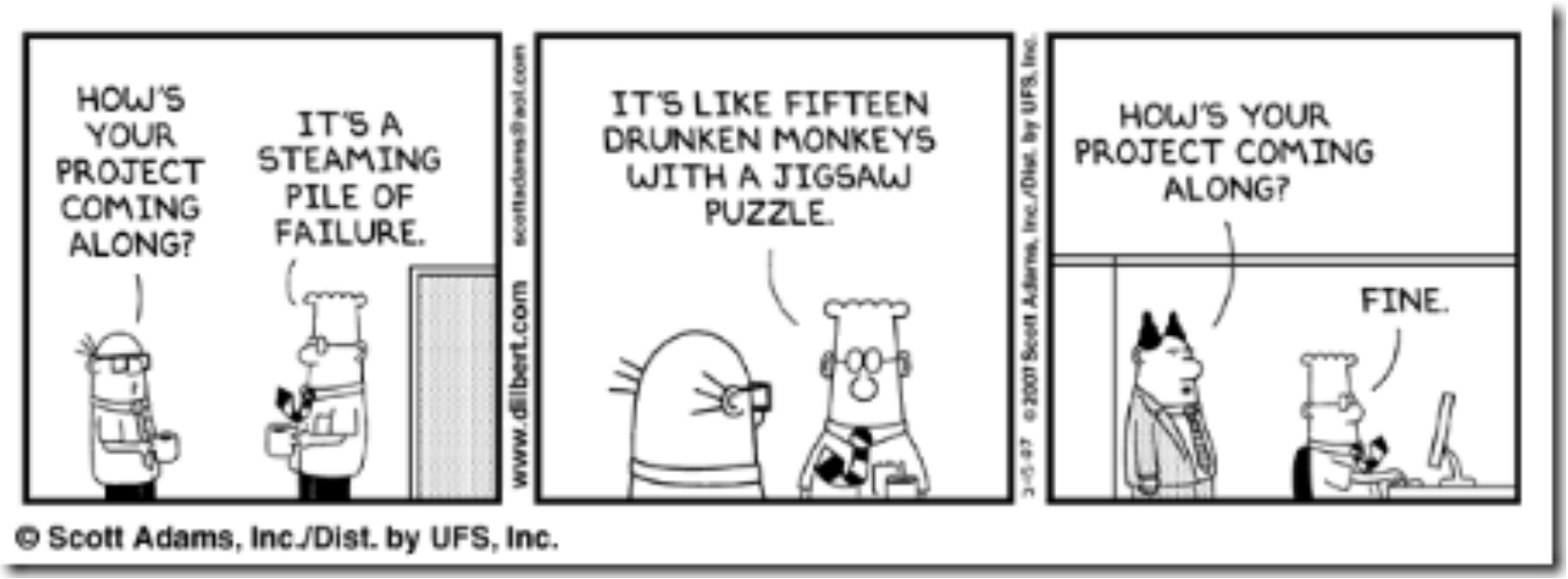
All the variables you evaluate when making a move across the software lifecycle...

Risk Management

- More precisely risk **avoidance**
- Connected to hazards
  - A Hazard is any **real or potential** condition that can **cause injury**, illness, or death to personnel; damage to or loss of a system, equipment or resource; or damage to the environment.

Definition of Risk

- A risk is a potential problem – it might happen and it might not
- Conceptual definition of risk
  - Risk concerns future happenings
  - Risk involves change in mind, opinion, actions, places, etc.
  - Risk involves choice and the uncertainty that choice entails
- Two characteristics of risk
  - Uncertainty – the risk may or may not happen, that is, there are no 100% risks (those, instead, are called constraints)
  - Loss – the risk becomes a reality and unwanted consequences or losses occur


HOW'S YOUR PROJECT COMING ALONG?
IT'S A STEAMING PILE OF FAILURE.
www.dilbert.com scottadams@aol.com
IT'S LIKE FIFTEEN DRUNKEN MONKEYS WITH A JIGSAW PUZZLE.
3-5-97 ©2007 Scott Adams, Inc./Dist. by UFS, Inc.
HOW'S YOUR PROJECT COMING ALONG?
FINE.
© Scott Adams, Inc./Dist. by UFS, Inc.

Top ten risk factors

- Personnel shortfall
- Unrealistic schedule/budget
- Wrong functionality
- Wrong user interface
- Goldplating
- Requirements volatility
- Bad external components
- Bad external tasks
- Real-time shortfalls
- Capability shortfalls

Risk Categorization – Approach #1

- Project risks
  - They threaten the project plan
  - If they become real, it is likely that the project schedule will slip and that costs will increase
- Technical risks
  - They threaten the quality and timeliness of the software to be produced
  - If they become real, implementation may become difficult or impossible
- Business risks
  - They threaten the viability of the software to be built
  - If they become real, they jeopardize the project or the product

(More on next slide)

Risk Categorization – Approach #1 (continued)

- Sub-categories of Business risks
  - **Market risk** – building an excellent product or system that no one really wants
  - **Strategic risk** – building a product that no longer fits into the overall business strategy for the company
  - **Sales risk** – building a product that the sales force doesn't understand how to sell
  - **Management risk** – losing the support of senior management due to a change in focus or a change in people
  - **Budget risk** – losing budgetary or personnel commitment

Risk Categorization – Approach #2

- Known risks
  - Those risks that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date)
- Predictable risks
  - Those risks that are extrapolated from past project experience (e.g., past turnover)
- Unpredictable risks
  - Those risks that can and do occur, but are extremely difficult to identify in advance

Reactive vs. Proactive Risk Strategies

- Reactive risk strategies
  - "Don't worry, I'll think of something"
  - The majority of software teams and managers rely on this approach
  - Nothing is done about risks until something goes wrong
    - The team then flies into action in an attempt to correct the problem rapidly (fire fighting)
  - Crisis management is the choice of management techniques
- Proactive risk strategies
  - Steps for risk management are followed (see next slide)
  - Primary objective is to avoid risk and to have a contingency plan in place to handle unavoidable risks in a controlled and effective manner

Steps for Risk Management

1) Identify possible risks; recognize what can go wrong
2) Analyze each risk to estimate the probability [L,M,H] that it will occur and the impact (i.e., damage) that it will do if it does occur
3) Rank the risks by probability and impact
   - Impact may be negligible, marginal, critical, and catastrophic
4) Develop a contingency plan to manage those risks having high probability and high impact

Risk types and examples

Risk	Probability	Effects
Organizational financial problems force reductions in the project budget.	Low	Catastrophic
It is impossible to recruit staff with the skills required for the project.	High	Catastrophic
Key staff are ill at critical times in the project.	Moderate	Serious
Faults in reusable software components have to be repaired before these components are reused.	Moderate	Serious
Changes to requirements that require major design rework are proposed	Moderate	Serious
The organization is restructured so that different management are responsible for the project	High	Serious
The database used in the system cannot process as many transactions per second as expected	Moderate	Serious

Strategies to help manage risk: examples

Risk	Strategy
Organizational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business and presenting reasons why cuts to the project budget would not be cost-effective.
Recruitment problems	Alert customer to potential difficulties and the possibility of delays; investigate buying-in components.
Staff illness	Reorganize team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.
Requirements changes	Derive traceability information to assess requirements change impact; maximize information hiding in the design.

Strategies to help manage risk: examples

Risk	Strategy
Organizational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying-in components; investigate use of a program generator.

Known and Predictable Risks

- **Product size** – risks associated with overall size of the software to be built
- **Business impact** – risks associated with constraints imposed by management or the marketplace
- **Customer characteristics** – risks associated with sophistication of the customer and the developer's ability to communicate with the customer in a timely manner
- **Process definition** – risks associated with the degree to which the software process has been defined and is followed
- **Development environment** – risks associated with availability and quality of the tools to be used to build the project
- **Technology to be built** – risks associated with complexity of the system to be built and the "newness" of the technology in the system
- **Staff size and experience** – risks associated with overall technical and project experience of the software engineers who will do the work

Seven Principles of Risk Management

- **Maintain a global perspective**
  - View software risks within the context of a system and the business problem that is is intended to solve
- **Take a forward-looking view**
  - Think about risks that may arise in the future; establish contingency plans
- **Encourage open communication**
  - Encourage all stakeholders and users to point out risks at any time
- **Integrate risk management**
  - Integrate the consideration of risk into the software process
- **Emphasize a continuous process of risk management**
  - Modify identified risks as more becomes known and add new risks as better insight is achieved
- **Develop a shared product vision**
  - A shared vision by all stakeholders facilitates better risk identification and assessment
- **Encourage teamwork when managing risk**
  - Pool the skills and experience of all stakeholders when conducting risk management activities

Managing People is Key!

- Software engineering means people, not things
- Civil Engineering is (arguably) easier to manage (less uncertainty)
- People need to be part of the whole, they are skilled professionals not cogs in a machine
- People can become risks as well, e.g., employee turnover, organizational rebellion
- Consistently create a group or community

Activities involved in managing people

- Selecting staff
- Motivating people
- Managing groups

People management factors

- Consistency
  - Team members should all be treated in a comparable way without favourites or discrimination.
- Respect
  - Different team members have different skills and these differences should be respected.
- Inclusion
  - Involve all team members and make sure that people's views are considered.
- Honesty
  - You should always be honest about what is going well and what is going badly in a project.

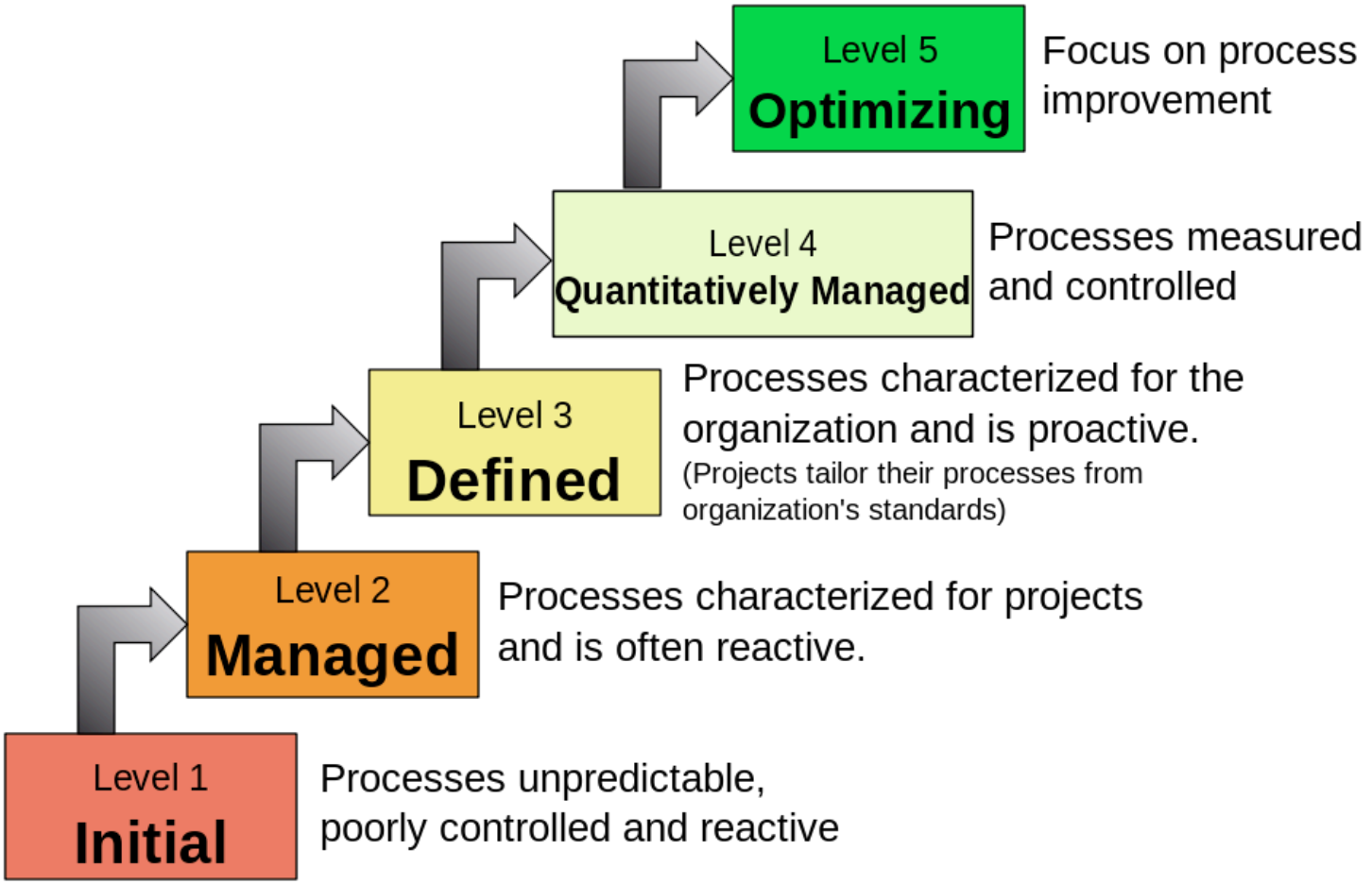
Roadmap

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

Project Managers, you'll never walk alone: Capability Maturity Model & Integration

- What is CMMI?
  - "it is a process improvement training and appraisal program, it is required by many DOD and U.S. Government contracts" [Thanks to Wikipedia ☺]
  - Exists for various areas of interest (AOI) of software engineering
  - We focus on *Development AOI*
  - CMMI Institute, from CMU-SEI (Carnegie-Mellon University, Software Engineering Institute)

Essentially: Let me see what you do and I'll tell you how good you are...



Behaviors at the Five Levels

Maturity Level	Process Characteristics	Behaviors
Optimizing	Focus is on continuous quantitative improvement	Focus on "fire prevention"; improvement anticipated and desired, and impacts assessed.
Quantitatively Managed	Process is measured and controlled	Greater sense of teamwork and inter-dependencies
Defined	Process is characterized for the organization and is proactive	Reliance on defined process. People understand, support and follow the process.
Managed	Process is characterized for projects and is often reactive	Over reliance on experience of good people – when they go, the process goes. "Heroics."
Initial	Process is unpredictable, poorly controlled, and reactive	Focus on "fire fighting"; effectiveness low – frustration high.

Maturity Level 1
Initial

- Maturity Level 1 deals with performed processes.
- Processes are unpredictable, poorly controlled, reactive.
- The process performance may not be stable and may not meet specific objectives such as quality, cost, and schedule, but useful work can be done.

Maturity Level 2
Managed at the Project Level

- Maturity Level 2 deals with managed processes.
- A managed process is a performed process that is also:
  - Planned and executed in accordance with policy
  - Employs skilled people
  - Adequate resources are available
  - Controlled outputs are produced
  - Stakeholders are involved
  - The process is reviewed and evaluated for adherence to requirements
- Processes are planned, documented, performed, monitored, and controlled at the project level. Often reactive.
- The managed process comes closer to achieving the specific objectives such as quality, cost, and schedule.

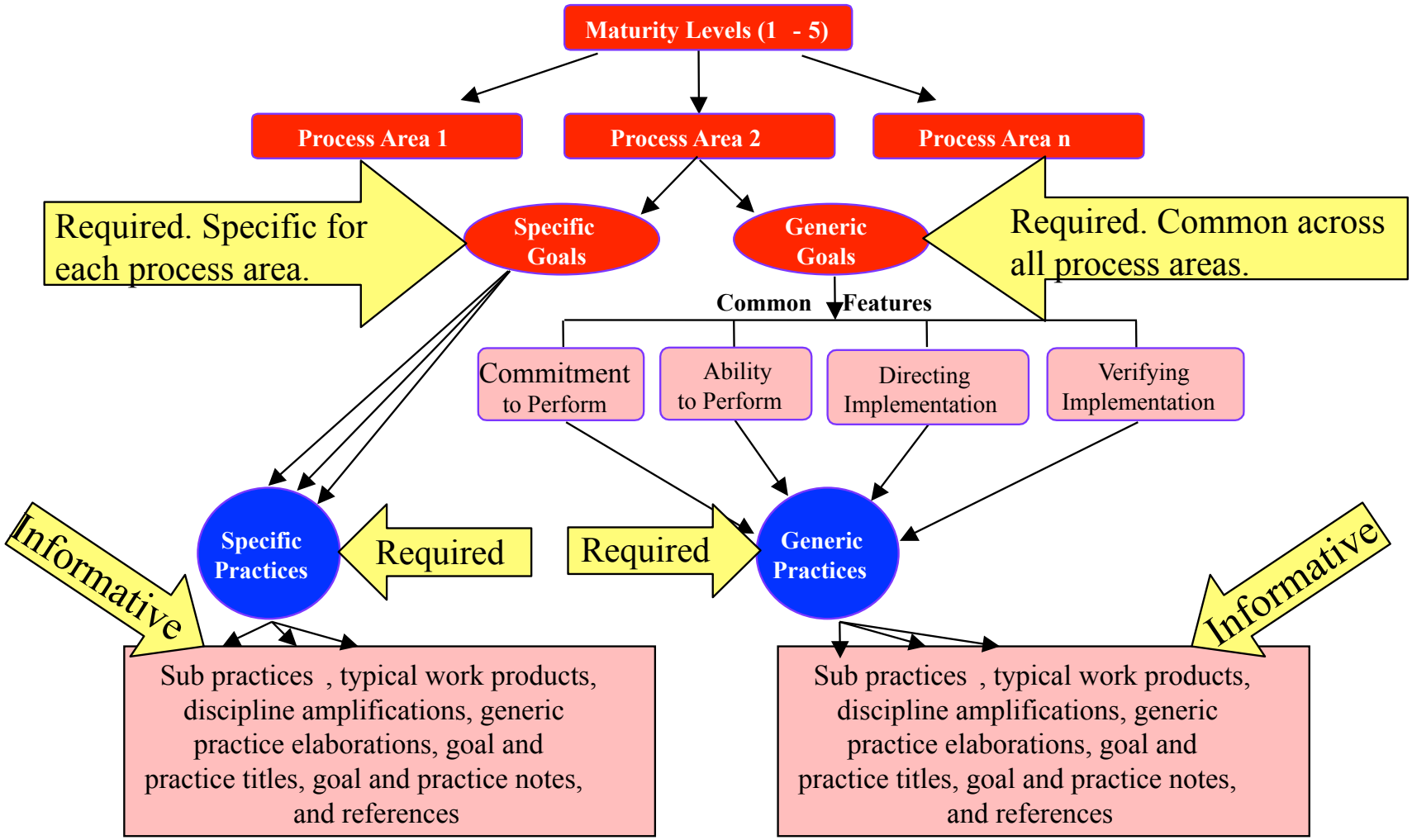
Maturity Level 3
Defined at the Organization Level

- Maturity Level 3 deals with defined processes.
- A defined process is a managed process that:
  - Well defined, understood, deployed and executed across the entire **organization**. Proactive.
  - Processes, standards, procedures, tools, etc. are defined at the organizational (Organization X) level. Project or local tailoring is allowed, however it must be based on the organization's set of standard processes and defined per the organization's tailoring guidelines.
- Major portions of the organization cannot "opt out."

CMMI Certification Components

- Within each of the 5 Maturity Levels, there are basic functions that need to be performed – these are called **Process Areas (PAs).**
- For Maturity Level 2 there are 7 Process Areas that must be completely satisfied.
- For Maturity Level 3 there are 11 Process Areas that must be completely satisfied.
- Given the interactions and overlap, it becomes more efficient to work the Maturity Level 2 and 3 issues concurrently.
- Within each PA there are **Goals** to be achieved and within each Goal there are **Practices**, work products, etc. to be followed that will support each of the Goals.

CMMI Terminology & Structure



Example

For the Requirements Management Process Area:

An example **Goal** (required):

"Manage Requirements"

An example **Practice** to support the Goal (required):

"Maintain bi-directional traceability of requirements"

Examples (suggested, but not required) of typical **Work Products** might be

Requirements traceability matrix or

Requirements tracking system

Yet another CMMI term: Institutionalization

- This is the most difficult part of CMMI implementation and the portion where managers play the biggest role and have the biggest impact
- Building and reinforcement of corporate culture that supports methods, practices and procedures so they are the ongoing way of business……..
  - Must be able to demonstrate institutionalization of all CMMI process areas for all organizations, technologies, etc.
- Required for all Process Areas

How can CMMI help?

- CMMI provides a way to focus and manage hardware and software development from product inception through deployment and maintenance.
  - ISO/TL9000 are still required. CMMI interfaces well with them. CMMI and TL are complementary - both are needed since they address different aspects.
    - ISO/TL9000 is a process compliance standard
    - CMMI is a process improvement model
- Behavioral changes are needed at both management and staff levels. Examples:
  - Increased personal accountability
  - Tighter links between Product Management, Development, SCN, etc.
- Initially a lot of investment required – but, if properly managed, we will be more efficient and productive while turning out products with consistently higher quality.

- Any Questions?

References

- Ian Sommerville, Software Engineering, Pearson (currently at the 10th edition)
- Function Points User Group http://www.ifpug.org/
- COCOMO II – Model Definition Manual http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf