CSIT214/CSIT883
IT Project Management

Project cost and effort management

Project management framework (review)

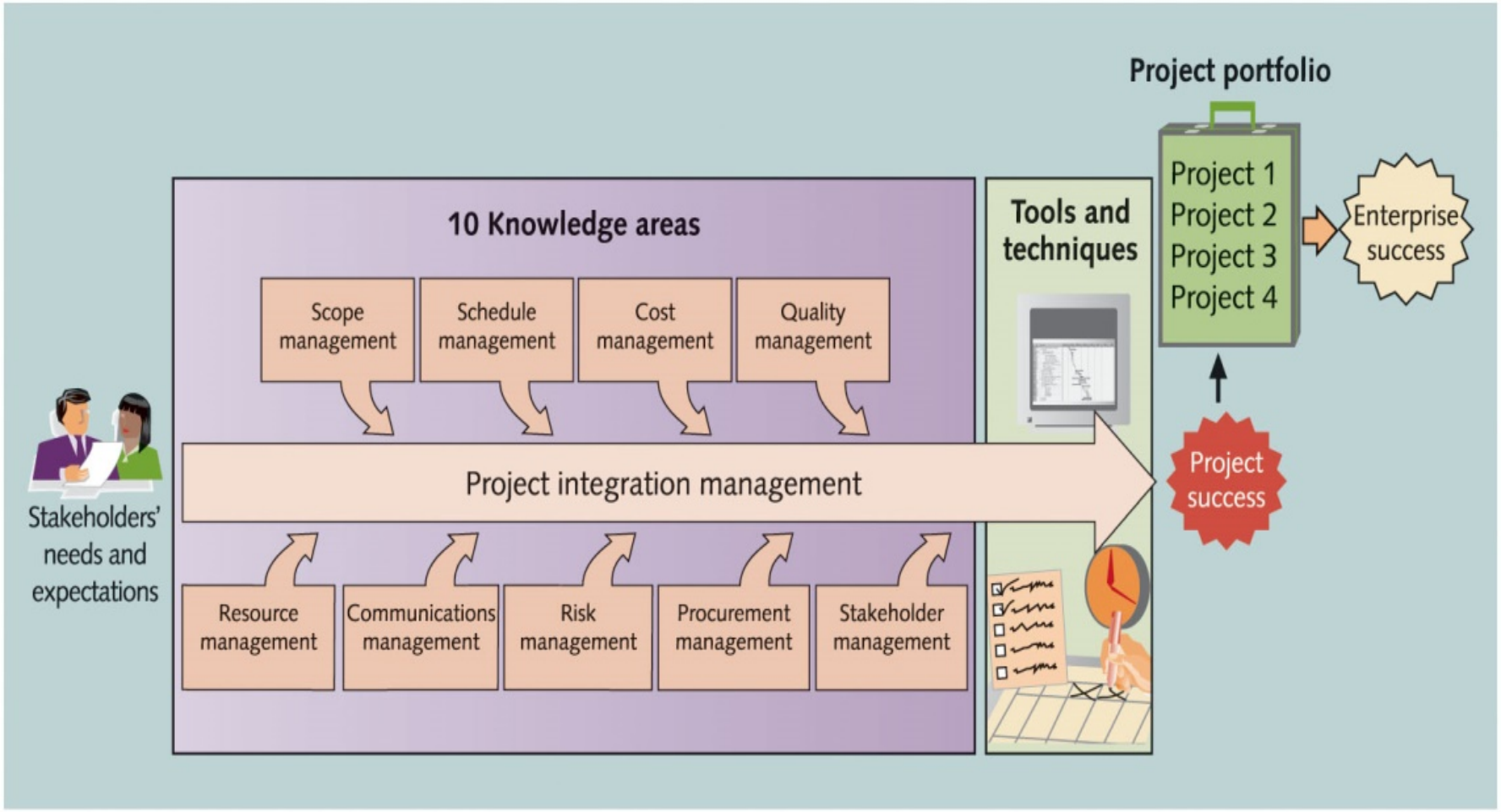


FIGURE 1-2 Project management framework

The Importance of Project Cost Management

- IT projects have a poor track record for meeting budget goals
  - Cost overrun is the additional percentage or dollar amount by which actual costs exceed estimates
  - A 2011 *Harvard Business Review* study reported an average cost overrun of 27 percent
    - Most important finding was the discovery of a large number of gigantic overages or "black swans"; a high-impact event that is rare and unpredictable, but not improbable in retrospect

What Went Wrong?

- The United Kingdom's National Health Service IT modernization program was called the greatest IT disaster in history with an estimated $26 billion overrun
  - Program had problems due to incompatible systems, resistance from physicians, and arguments among contractors about who's responsible for what
  - Scrapped in 2011

What is Cost?

- Cost is a resource sacrificed or foregone to achieve a specific objective or something given up in exchange
  - Usually measured in monetary units like dollars that must be paid to acquire goods and services

What is Project Cost Management? (1 of 2)

- Project cost management includes the processes required to ensure that the project is completed within an approved budget
  - **Planning cost management**: determining the policies, procedures, and documentation that will be used for planning, executing, and controlling project cost
  - **Estimating costs**: developing an approximation or estimate of the costs of the resources needed to complete a project
  - **Determining the budget**: allocating the overall cost estimate to individual work items to establish a baseline for measuring performance
  - **Controlling costs**: controlling changes to the project budget

Basis for successful estimating

- Information about past projects
  - Need to collect performance details about past project: how big were they? How much effort/time did they need? What is the team's productivity?
- Need to be able to measure the amount of work involved
  - Traditional size measurement for software is 'lines of code' .

A taxonomy of estimating methods

- Bottom-up: activity based, analytical
- Parametric or algorithmic models (top-down)
  - e.g. function points, COCOMO
- Expert opinion - just guessing?
- Analogy - case-based, comparative

Bottom-up estimating

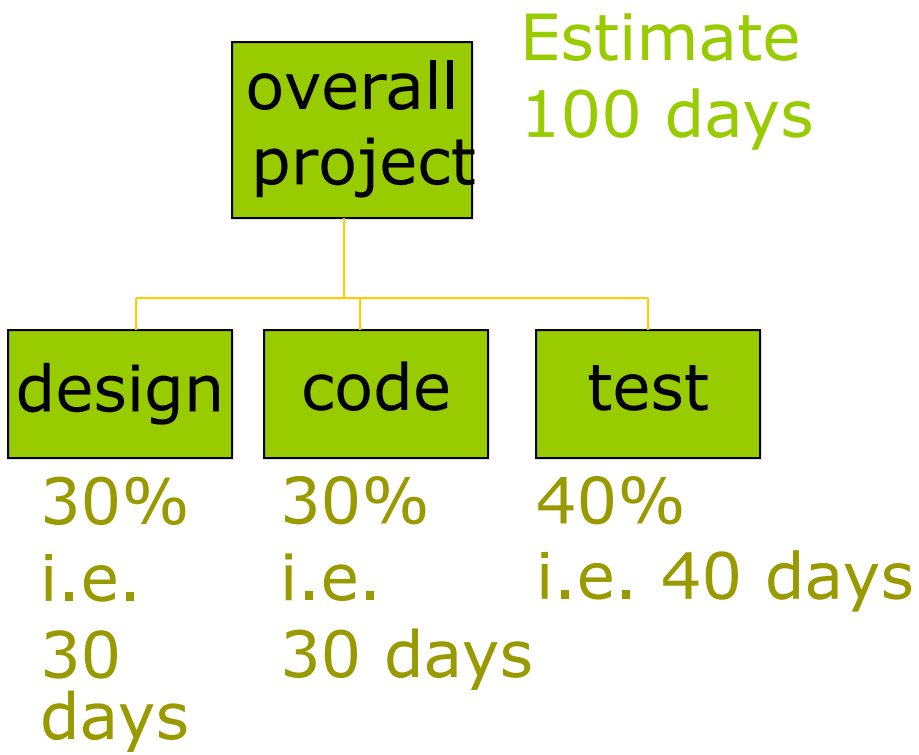
1. Break project into smaller and smaller components

[2. Stop when you get to what one person can do in one/two weeks]

3. Estimate costs for the lowest level activities

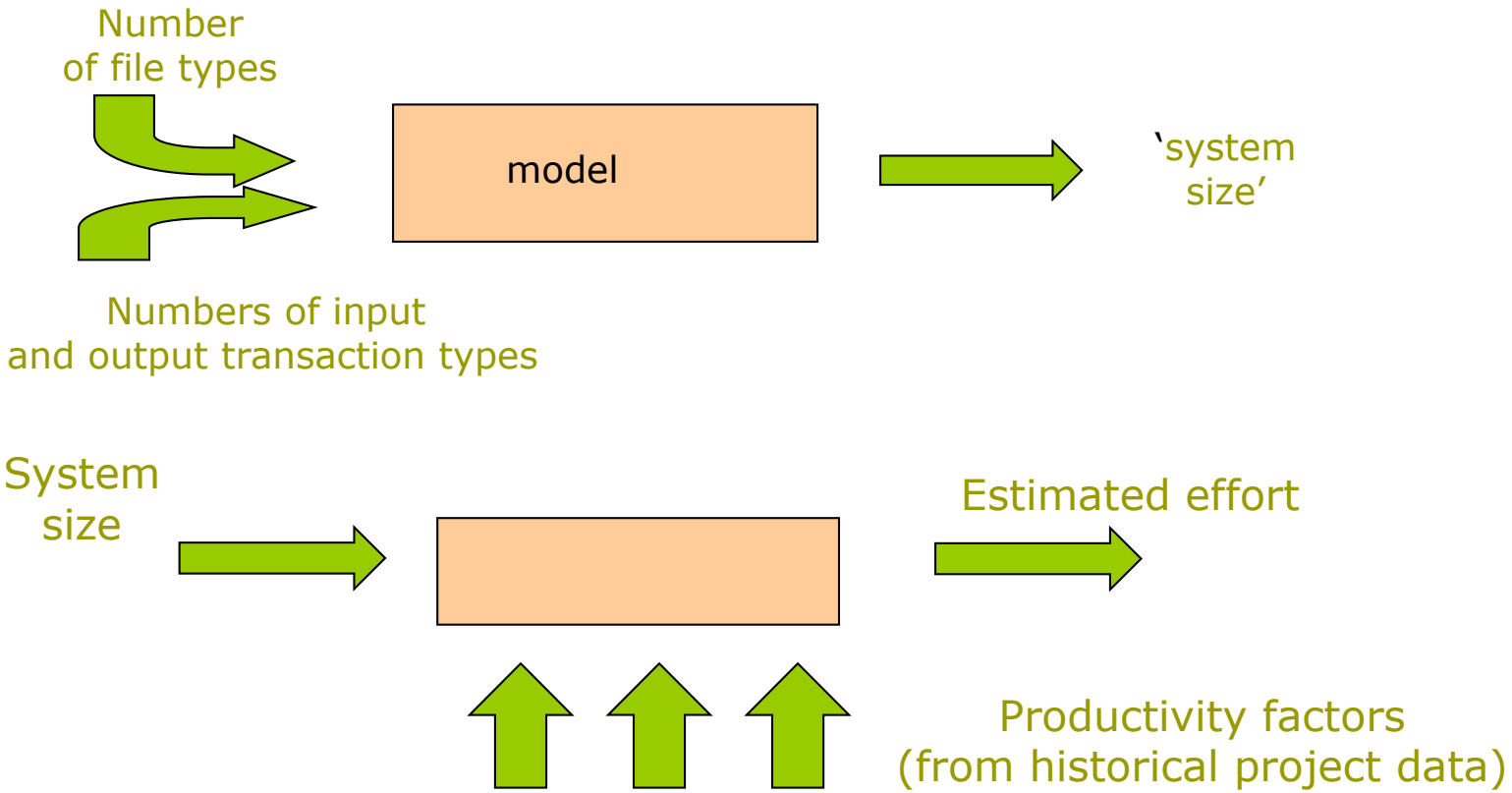
4. At each higher level calculate estimate by adding estimates for lower levels

Top-down estimates



- Produce overall estimate using effort driver(s)
- Distribute proportions of overall estimate to components

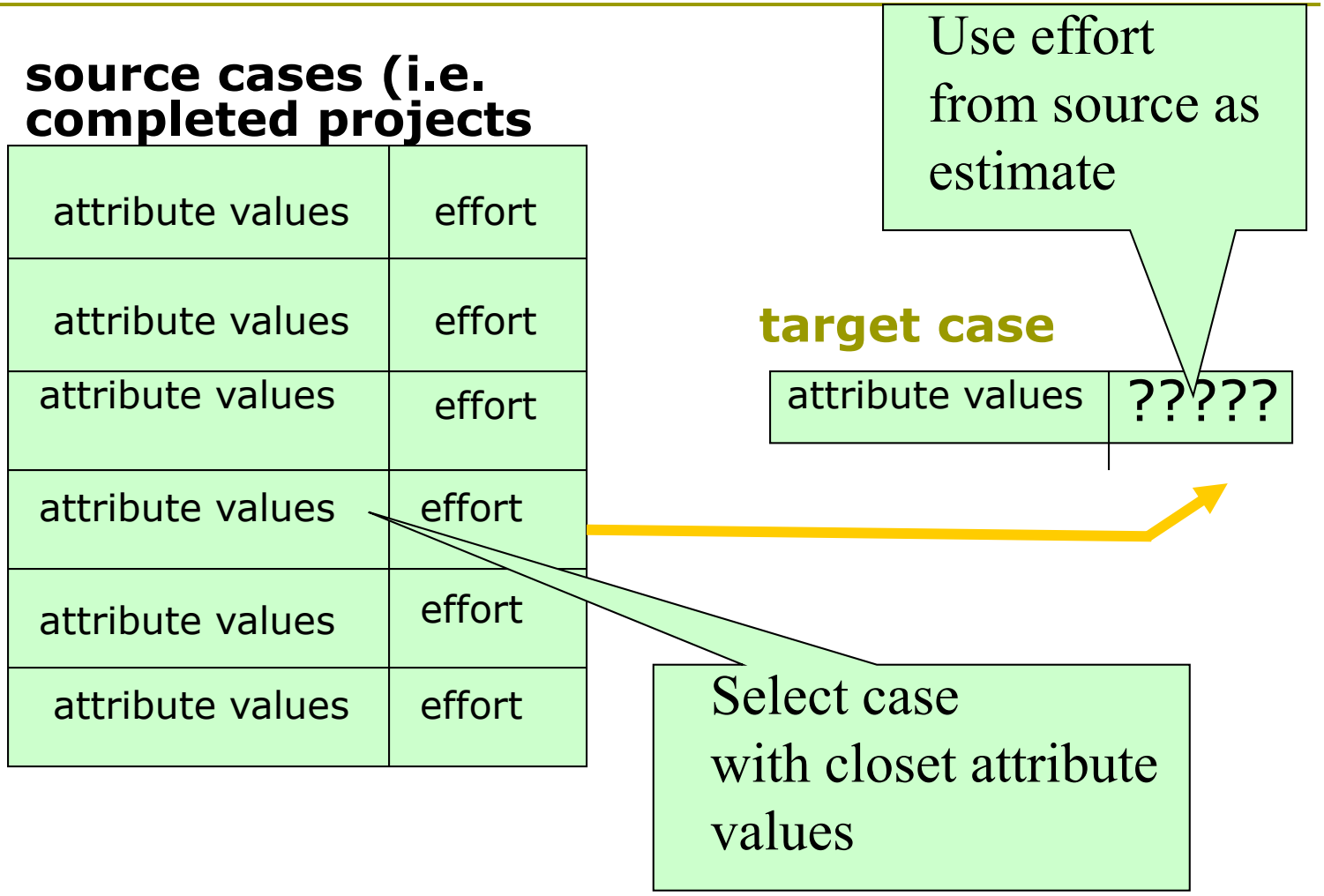
Algorithmic/Parametric models



Expert judgement

- Asking someone who is familiar with and knowledgeable about the application area and the technologies to provide an estimate
- Particularly appropriate where existing code is to be modified (i.e. change impact analysis)
- Research shows that experts judgement in practice tends to be based on analogy

Estimating by analogy (case-based reasoning)



Parametric models

We are now looking more closely at parametric models:

1. Albrecht/IFPUG function points
2. COCOMO81 and COCOMO II

Albrecht/IFPUG function points

- Developed by Allan Albrecht in 1979 at IBM
- A large user group led by the International Function Point Users Group (IFPUG http://www.ifpug.org)
- The original function point counting technique was refined over time. Latest version is IPUG's Function Point Counting Practices Manual 4.3 (Released in 2010)
- It has became an ISO standard.

Albrecht/IFPUG function points - continued

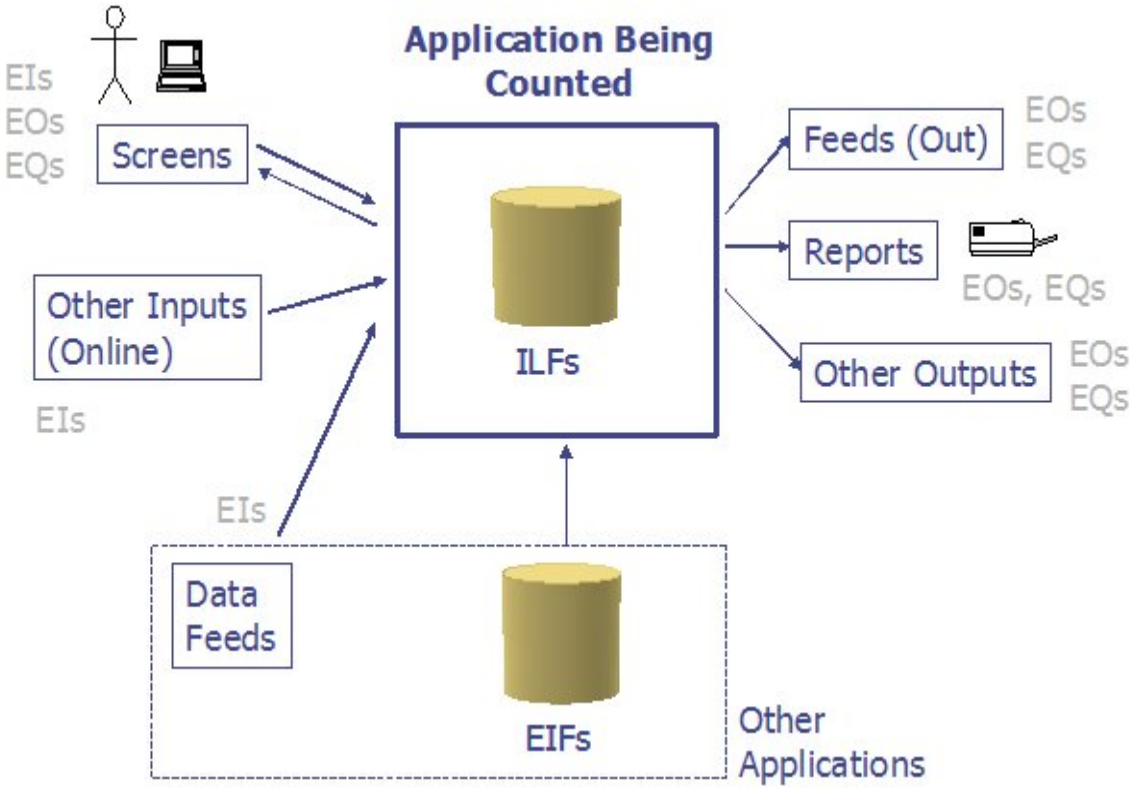
- Five function types and are classified into 2 groups:
  - **Data Functions:**
    - **Internal Logical Files (ILFs):** represent user-identifiable data that is stored **within** your application.
      - E.g. Tables in a relational databases, flat files.
    - **External Interface Files (EIFs):** represent the data that your application will **use/reference** but the data is **not** maintained by your application.
      - EIFs is the list of ILFs maintained by other applications.

cont…

Albrecht/IFPUG function points - continued

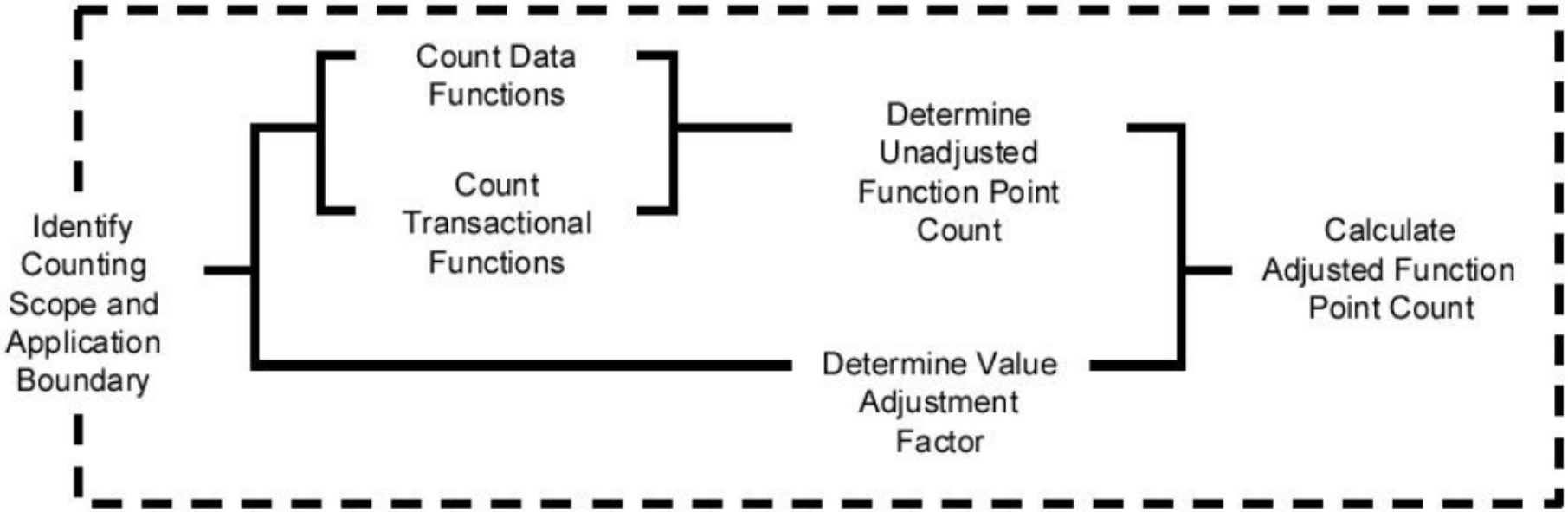
- Transaction Functions:
  - **External Inputs (EIs):** input transactions which **update** ILFs.
    - E.g. Data entry by users, data or file feeds by external applications.
  - **External Outputs (EOs):** transactions which **extract and display** data from ILFs.
    - E.g. Reports created by your application where the reports include *derived* information (i.e. **your application needs to do some computation**)
  - **External Queries (EQs):** user initiated transactions which provide information but do **not update** ILFs.
    - E.g. Reports created by your application but the reports do **not** contain any *derived* data. (i.e. your application does **not** do any computation)

Albrecht/IFPUG function points - continued



- Data functions:
  - Internal Logical Files (ILFs)
  - External Interface Files (EIFs)
- Transactional functions:
  - External Inputs (EIs)
  - External Outputs (EOs)
  - External Queries (EQs)

How to count function points



Albrecht complexity multipliers

	Low complexity	Medium complexity	High complexity
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Examples

Payroll application has:

- A transaction of medium complexity to **input, amend and delete** employee details
  - an EI that is rated of medium complexity
- A transaction of high complexity that **calculates and updates** pay details from timesheet data that is input
  - an EI of high complexity
- A transaction of medium complexity **that computes and prints out** pay-to-date details for each employee
  - an EO of medium complexity
- A file of payroll details for each employee (medium complexity)
  - assessed as of medium complexity ILF
- A simple personnel file **maintained by another system** is accessed for name and address details
  - a simple, i.e. low-complexity, EIF

What would be the FP counts for these?

FP counts

	Low complexity	Medium complexity	High complexity
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

1. Medium EI — 4 FPs
2. High complexity EI — 6 FPs
3. Medium complexity EO — 5 FPs
4. Medium complexity LIF — 10 FPs
5. Simple EIF — 5 FPs

Total — **30 FPs**

- If previous projects delivered 5 FPs a day, implementing the above should take 30/5 = 6 days

Albrecht/IFPUG function points How to assess complexity?

- For data functions, the rating is based on:
  - RET: the number of **Record Element Types** (i.e. subgroup of data elements) in an ILF or EIF
    - E.g. a customer file that contains Name, Address, so on and so forth. In addition, all the credit cards and credit card numbers of the customer are contained in the file. Hence, there are two RETs in the Customer File.
  - DET: the number of **Data Element Types** (i.e. **unique, non-repeated** field) in an ILF or EIF.

RETS	Data Element Types (DETs)		
	1-19	20-50	51+
1	Low	Low	Medium
2 to 5	Low	Medium	High
6 or more	Medium	High	High

Example

- A internal logical file contains data about purchase orders. These orders are organized into two separate record types: the main Purchase-Order details (including purchase order number, supplier reference and purchase order date) and details for each Purchase-Order-Item specified in the order (including product code, unit price, and number ordered).
  - What is the number of record element types for this file?
  - What is the number of data element types?
  - What are the function point count for this file?

Albrecht/IFPUG function points How to assess complexity? (cont.)

- For transactional functions, the rating is based on:
  - FTR: the number of File Type References (ILFs or EIFs) in a transaction
  - DET: the number of Data Element Types in a transaction

FTRs	Data Element Types (DETs)		
	1-5	6-19	20+
0-1	Low	Low	Medium
2-3	Low	Medium	High
4 or more	Medium	High	High

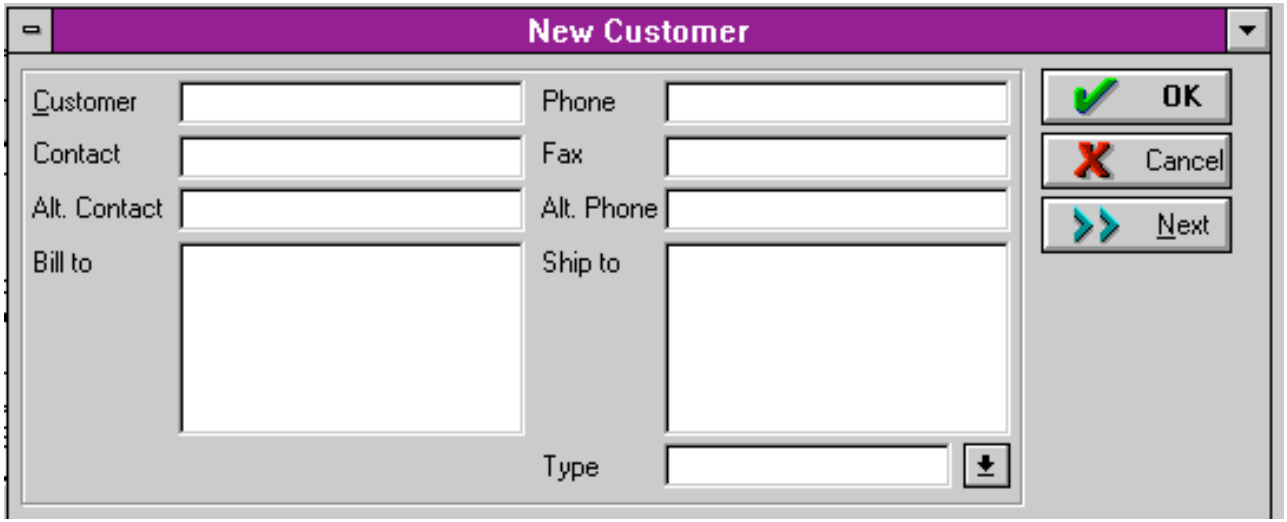
For EO and EQ

FTRs	Data Element Types (DETs)		
	1-4	5-15	16+
0-1	Low	Low	Medium
2	Low	Medium	High
3 or more	Medium	High	High

For EI

Pen and paper exercise

- How many data elements (DETs) are there in this input screen?
- If this screen updates one internal logical file (ILF) , how many function points does this screen represent?



Note:

1. *This screen is used to add a new customer to an application. The OK command button and the Next command button both add the new customer to the database.*
2. *In GUI applications, a data element (DET) is information that is stored on an internal logical file or that is used to invoke a transaction*

Pen and paper exercise

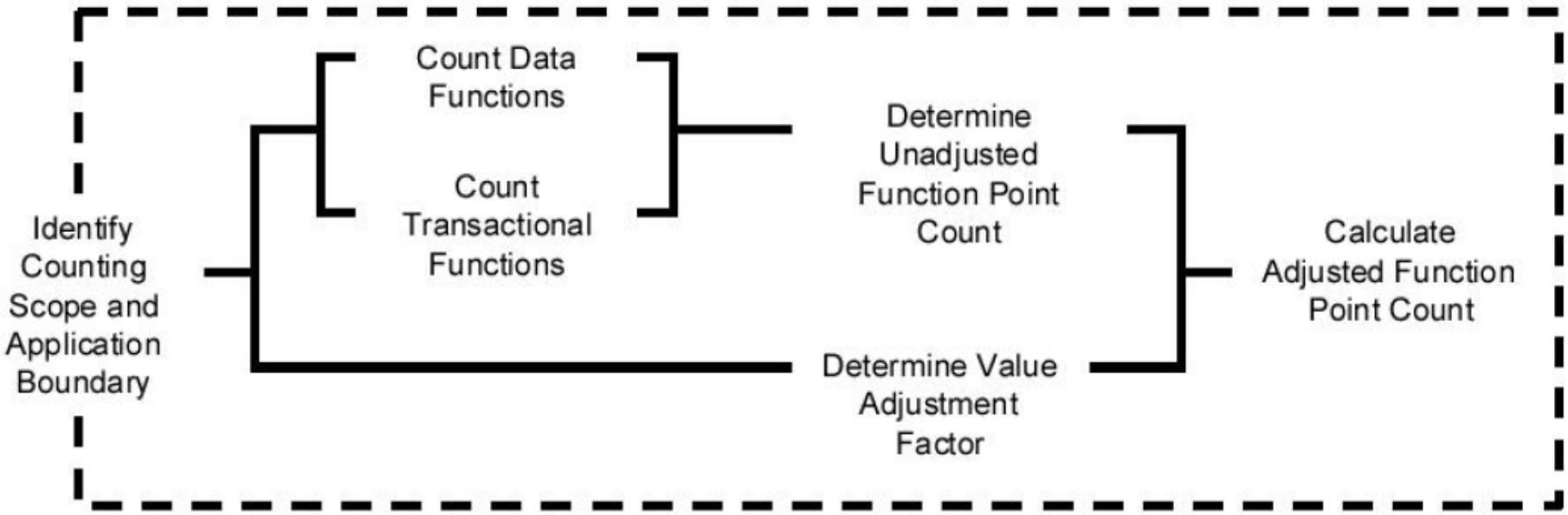
- Application A receives input from running a batch input file. The batch file is one physical file, but contains many different types of records. The first field is a record identifier number. The record identifier number can range from 1-75. The second field describes if the record is **new and adds** to the file, **changes** a previous batch input or a **deletes** a previous batch input (add, change and delete).

  Depending on the *record identifier number* there are a unique set of data elements, a different set of files are updated and referenced, and different processing logic is followed.

  Every single record identifier number updates more than 3 files (has more than 3 FTR's) and contains more than 5 data elements.

How many function points does this one batch input represent?

How to count function points Determine Value Adjustment Factor



- Previously, we have computed the **UFP (Unadjusted Function Points)**
- The last step involves assessing **the environment and processing complexity of the application** as a whole.
- In this step, the impact of 14 general system characteristics is **rated on a scale from 0 to 5** in terms of their likely effect on the project or application

Determine Value Adjustment Factor

General System Characteristic		Brief Description
1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Did the user require response time or throughput?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF's are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Determine Value Adjustment Factor (VAF)

- The calculation of VAF is based on the TDI (**Total Degree of Influence** of the 14 General system characteristics)
  - *TDI = Sum of (DI of 14 General System Characteristics) where DI stands for Degree of Influence.*
  - *VAF = 0.65 + (0.01 * TDI)*
- Finally the Adjusted Function Points or Function Points are
  - **FP = UFP * VAF**
    - where UFP is Unadjusted Function Points.

Example

Function Type	Estimated Count	Weight	FP-Count
EI	24	(Average) 4	96
EO	16	(Average) 5	80
EQ	22	(Average) 4	88
ILF	4	(Average) 10	40
ELF	2	(Average) 7	14
UFP count			**318**

Note: Average = Medium

- VAF = 52 * 0.01 + 0.65
  = 1.17
- FP = 318 x 1.17
  = 372

General System Characteristics (GSCs)	Degree of Influence (DI) 0 - 5
1. Data Communications	2
2. Distributed Data Processing	0
3. Performance	5
4. Heavily Used Configuration	5
5. Transaction Rate	2
6. Online Data Entry	4
7. End-User Efficiency	3
8. Online Update	5
9. Complex Processing	4
10. Reusability	5
11. Installation Ease	4
12. Operational Ease	3
13. Multiple Sites	4
14. Facilitate Change	5
Total Degree of Influence (TDI)	52
Value Adjustment Factor (VAF)	1.17

Pen and paper exercise

1. What is the value adjustment factor if all of the general system characteristics scored a value of 5 (strong influence)?
2. An application has a base unadjusted function point count of 500, a value adjustment factor of 1.10. What is the adjusted function point count?
3. An application has the following: 10 Low External Inputs, 12 High External Outputs, 20 Low Internal Logical Files, 15 High External Interface Files, 12 Medium External Queries, and a value adjustment factor of 1.10.
   - What is the unadjusted function point count?
   - What is the adjusted function point count?

Constructive Cost Model (COCOMO)

- Developed by Barry Boehm first in 1981 (COCOMO 81).
  - Latest version is COCOMO II (published in 2000)
- **Basic** model

  $$\textbf{effort} = \textbf{c} \times \textbf{size}^{\textbf{k}}$$

  - **c** and **k** depend on the type of system: organic, semi-detached, embedded
  - Size is measured in **KLOC (i.e. thousands of lines of code)**
  - Effort is measured in person-months

Further source: Barry Boehm et al. *Software estimation with COCOMO II,* Prentice-Hall 2002

The COCOMO constants

System type	c	k
Organic (broadly, information systems)	2.4	1.05
Semi-detached	3.0	1.12
Embedded (broadly, real-time)	3.6	1.20

- **Organic:** a small team develops a small system with flexible requirements in a highly familiar in-house environment
- **Embedded:** the system has to operate within very tight constraints and changes to the system very costly.
- **Semi-detached:** This combines elements of the organic and the embedded types or has characteristics that came between the two.

Example: what is the estimated effort of developing an organic system with 50,000 lines of code?

$$\textbf{effort} = c \times \textbf{size}^{k}$$

COCOMO II

An updated version of COCOMO:

- There are different COCOMO II models for estimating at the 'early design' stage and the 'post architecture' stage when the final system is implemented. We'll look specifically at the first.
- The core model is:

  $pm = A(size)^{(sf)} \times(em_1) \times(em_2) \times(em_3)....$

  where:

  pm = person months,

  A is 2.94,

  size is number of thousands of lines of code,

  sf is the scale factor, and

  em is an effort multiplier

COCOMO II Scale factor

Based on five factors which appear to be particularly sensitive to system size

1. Precedentedness (**PREC**): degree to which there are past examples that can be consulted
2. Development flexibility (**FLEX**): degree of flexibility that exists when implementing the project
3. Architecture/risk resolution (**RESL**): degree of uncertainty about requirements
4. Team cohesion (**TEAM**): degree to which there is a large dispersed team (e.g. in different locations) as opposed to there being a small tightly knit team.
5. Process maturity (**PMAT**): degree to how structured and organized the way the software is produced.

COCOMO II Scale factor values

Driver	Very low	Low	Nominal	High	Very high	Extra high
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

Example of scale factor

- A software development team is developing an application which is very similar to previous ones it has developed.
  - PREC is very high (score 1.24).
- A very precise software engineering document lays down very strict requirements.
  - FLEX is very low (score 5.07).
- The good news is that these tight requirements are unlikely to change
  - RESL is high with a score 2.83.
- The team is tightly knit
  - TEAM has high score of 2.19
- Processes are informal
  - So PMAT is low and scores 6.24

Scale factor calculation

The formula for sf is

sf = B + 0.01 × Σ scale factor values

E.g. sf = **0.91** +0.01 × (1.24 + 5.07 + 2.83 + 2.19 + 6.24)
= 1.0857

If system contained 10 KLOC then

Estimated Effort = $2.94 \times 10^{1.0857}$
= 35.8 person months

Effort multipliers

As well as the scale factor effort multipliers are also assessed:

RCPX	Product reliability and complexity
RUSE	Reuse required
PDIF	Platform difficulty
PERS	Personnel capability
FCIL	Facilities available
SCED	Schedule pressure

Effort multipliers

	Extra low	Very low	Low	Nominal	High	Very high	Extra high
RCPX	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE			0.95	1.00	1.07	1.15	1.24
PDIF			0.87	1.00	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED		1.43	1.14	1.00	1.00	1.00	

Example

- Say that a new project is similar in most characteristics to those that an organization has been dealing for some time
- **except**
  - the software to be produced is exceptionally complex and will be used in a safety critical system.
    - Product reliability and complexity (RCPX) is very high.
  - the software will interface with a new operating system that is currently in beta status.
    - Platform difficulty (PDIF) is very high
  - to deal with this the team allocated to the job are regarded as exceptionally good, but do not have a lot of experience on this type of software.
    - Personnel experience (PREX) is ranked nominal.
    - Personal capability (PERS) is ranked extra high.

Example -continued

RCPX	very high	1.91
PDIF	very high	1.81
PERS	extra high	0.50
PREX	nominal	1.00

All other factors are nominal

Say unadjusted estimate is 35.8 person months

With effort multipliers this becomes

Adjusted Estimated Effort = 35.8 x 1.91 x 1.81 x 0.5

= 61.9 person months

Pen and paper exercise

A new project has "average" novelty for the software supplier that is going to execute it and is thus given a nominal rating on this account for precedentedness. Development flexibility is high, but requirements may change radically and so the risk resolution exponent is rated very low. The development team area all located in the same office and this leads to team cohesion being rated as very high. The software company as a whole tends to be very informal in its standards and procedures, and the process maturity driver has therefore been given a rating of low.

- What would be the scale factor (sf) in this case?
- What would be the (unadjusted) estimate of effort of the size of the application was estimated as around 2000 lines of code?