CMPS 411

Software Project Estimation

Please read
Chapter 27 & 28

Dr. Abdelkarim Erradi

Dept. of Computer Science & Engineering

QU

Back

Outline

- Project Estimation
- Use Case Point Estimation method

Project Estimation

Estimating Methods We Can Use

Analogy - Compare the proposed project to previously completed similar projects where project development information is known.

Bottom-up - Identify and estimating each individual component separately, then combining the results to produce an estimate of the entire project.

Top-down - Project is partitioned into lower-level components and life cycle phases beginning at the highest level.

Expert Judgement - Human ‘experts’ consulted to provide an estimate based upon their experience and understanding of a proposed project.

Algorithmic - Use of equations from research and historical data to perform software estimates.

Estimating Software Project Size from Use Cases

- Estimating Software Projects (Size in Source Lines of Code (SLOC) and Effort (Person-Hours)) is difficult and often more art than science.
- The tools/methods that do exist require expert knowledge or experience
- Using Use Cases (UC) as a base for estimates
  - They represent the functional requirements of the to-be-developed software
  - In a a use case driven development process, UC provide a good base for top-down estimation
  - Allow obtaining estimates early for planning purposes, and then refine them iteration by iteration

Best practices for estimation

1. Combine estimates from different experts and estimation strategies.

2. Estimate top-down and bottom-up independently.

3. Justify and criticize estimates.

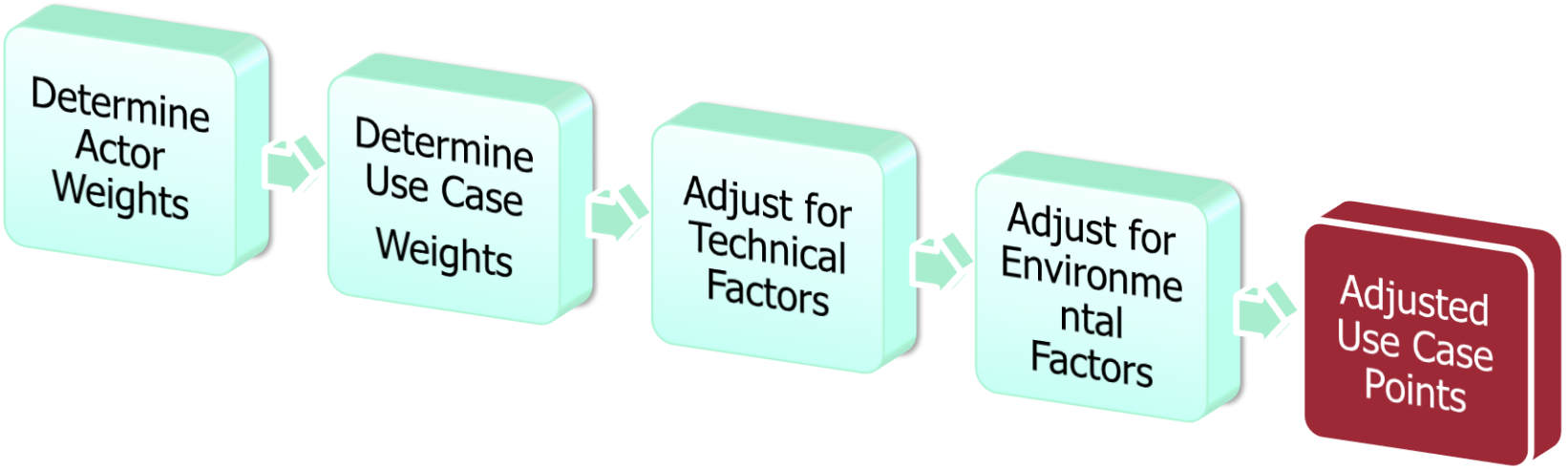
→ Use method based estimates to improve expert estimates.

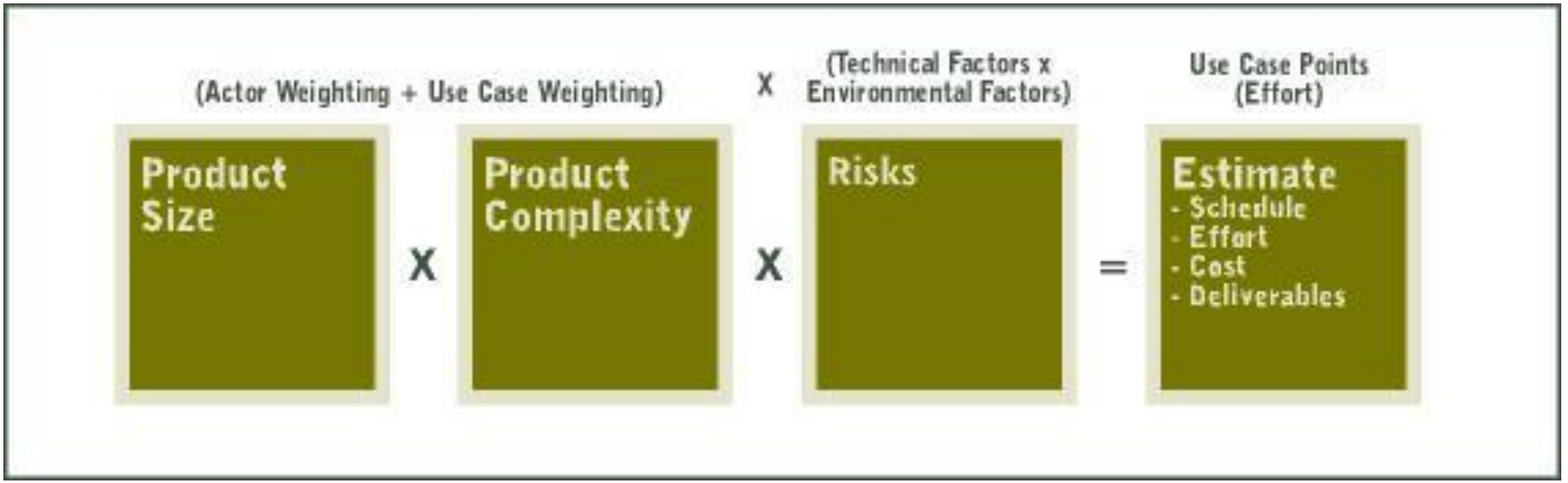
Use Case Point Estimation method

Use Case Point Estimation

- Each actor and use case is categorized according to **complexity** and **assigned a weight**.
  - The complexity of a use case is measured in number of transactions.
- The **unadjusted use case points** are calculated by adding the weights for each actor and use case.
- The unadjusted use case points are adjusted based on the values of **13 technical factors** and **8 environmental factors**.
- Finally the adjusted use case points are multiplied with a productivity factor.

Use Case Point Estimation – Process





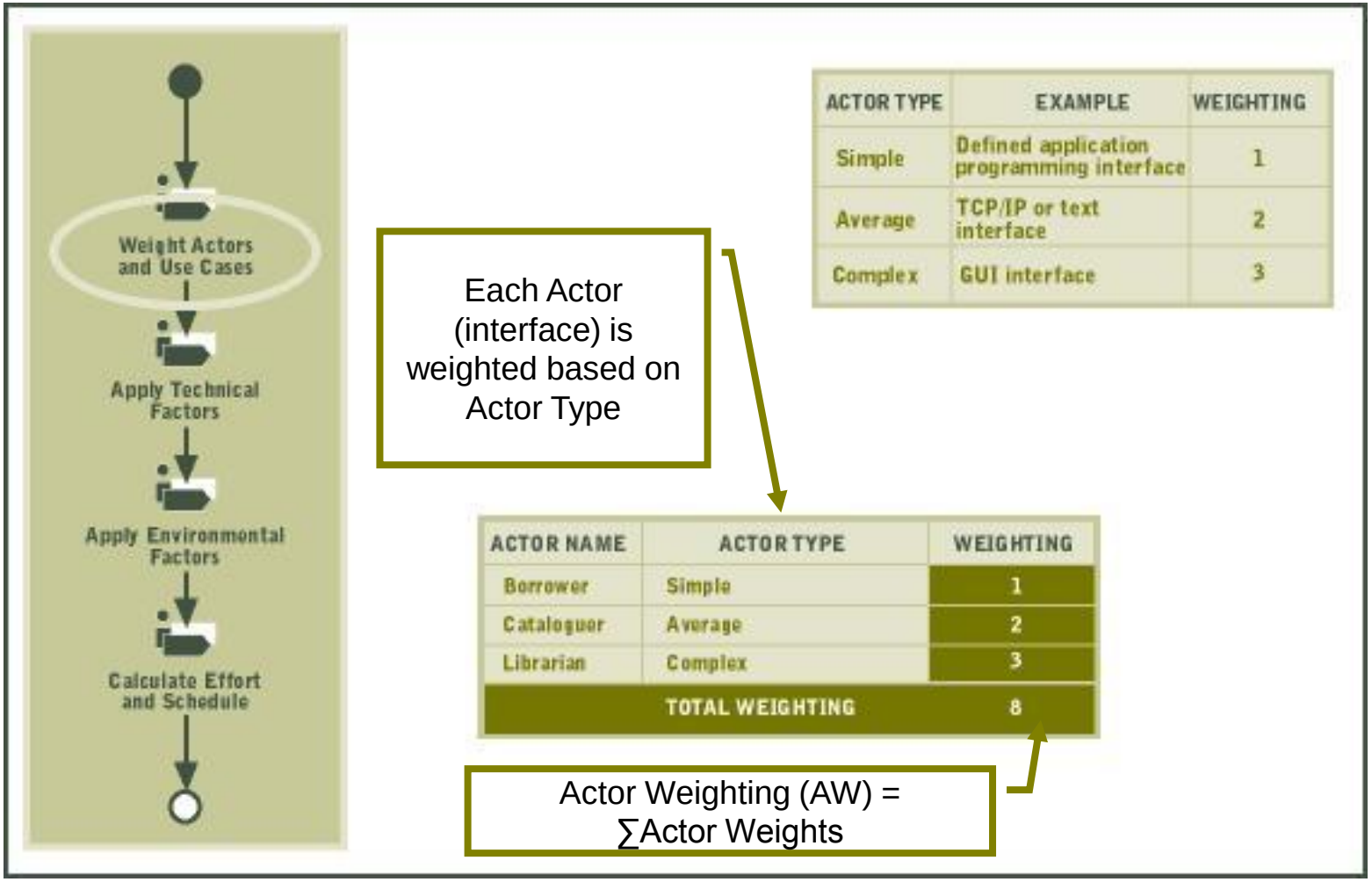
Actor Type	Description	Weight
Simple	The actor represents another system with a defined Application Programming Interface (API)	1
Average	The actor represents another system interacting through a protocol-driver Interface	2
Complex	The actor is a person interacting via a Graphical User Interface (GUI)	3

Unadjusted Actor Weight

- Actor initiates an interaction with the system to accomplish some goals.

- Does not represent the *physical* people or systems, but their *role*.

Weight Actors (AW)



ACTOR TYPE	EXAMPLE	WEIGHTING
Simple	Defined application programming interface	1
Average	TCP/IP or text interface	2
Complex	GUI interface	3

Each Actor (interface) is weighted based on Actor Type

ACTOR NAME	ACTOR TYPE	WEIGHTING
Borrower	Simple	1
Cataloguer	Average	2
Librarian	Complex	3
TOTAL WEIGHTING		8

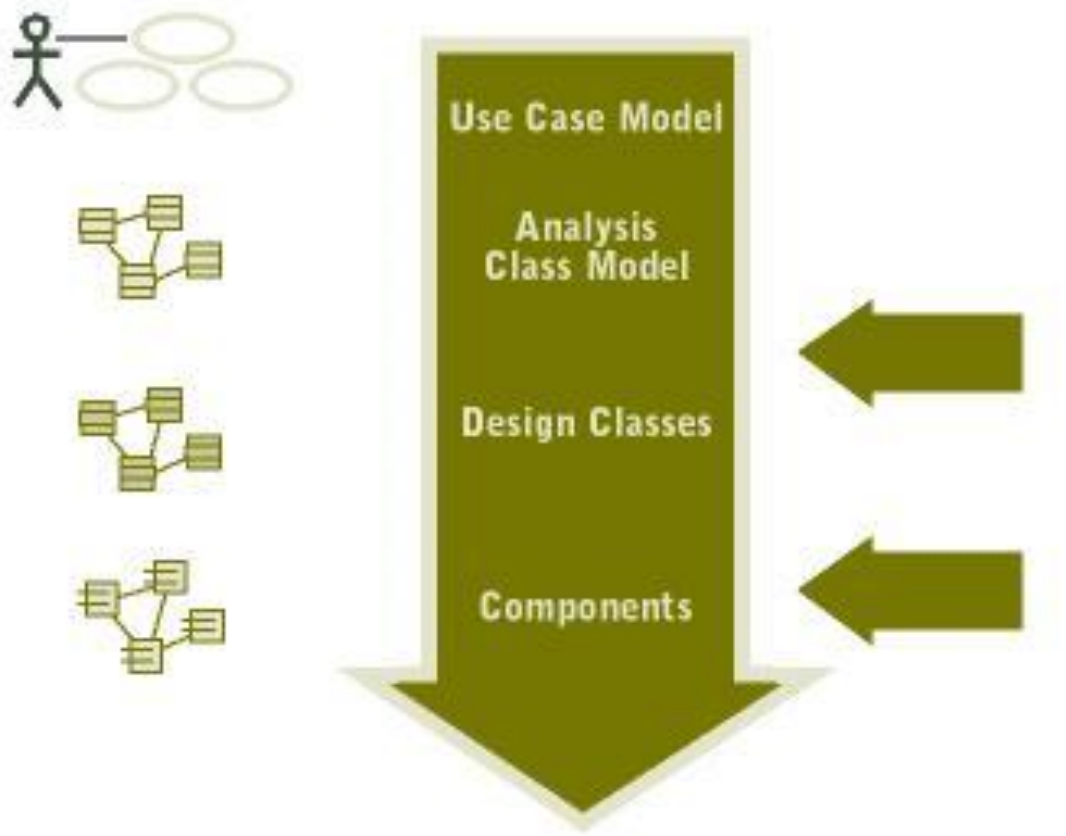
Actor Weighting (AW) = ∑Actor Weights

Use Case Type	Description	Weight
Simple	Less than 3 transactions	5
Average	4 to 7 transactions	10
Complex	More than 7 transactions	15

Unadjusted Use Case Weight

- Based on the total number of activities contained in all the use case scenarios

Product Complexity and Size



Product complexity is driven by architecture

Architecture is represented in many models!

Product Size is measured through artefacts derived from a Use Case

Weight Use Cases (UCW)

No of Flows (NoF)	WEIGHTING
NoF <4	5
<4 NoF <7	10
NoF >7	15

Use Case Weighting Factor is based on the Number of Use Case Flows

USE CASE NAME	No FLOWS	WEIGHTING FACTOR
Borrow item	4	10
Return item	3	5
Catalogue new item	3	5
Maintain borrower details	8	15
TOTAL WEIGHTING		35

Use Case Weighting (UCW) = ∑Use Case Weights

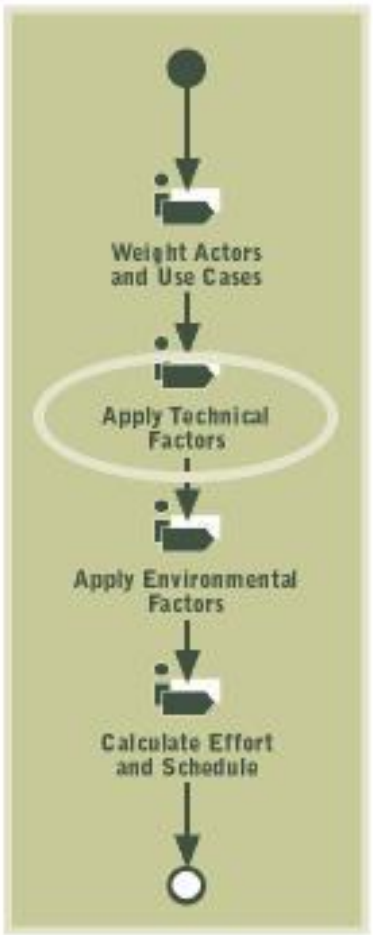
Adjust Based on Technical Factors

Factor number	Factor description	Weight
T1	Distributed system	2
T2	Response or throughput performance objective	1
T3	End-user efficiency	1
T4	Complex internal processing	1
T5	Code must be reusable	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Includes special security features	1
T12	Provides direct access for third parties	1
T13	Special user training facilities are required	1

- Each factor is assigned a value from 0 to 5 based on importance and how it is supported by the technology used.
- Technical Complexity Factor (TCF):

TCF = 0.6 + (0.01*Technical Total Factor)

Apply Technical Factors (TCF)



The Technical Complexity Factor (TCF) is derived by assigning a value to each technical factor on a scale of 0 to 5. 0 = Factor is irrelevant for this project
5 = Factor is essential.

Extended Value = Assigned Value x Weight

FACTOR DESCRIPTION	ASSIGNED VALUE	WEIGHT	EXTENDED VALUE
Distributed system	0	2	0
Response or throughput performance	5	1	5
End Use Efficiency	5	1	5
Complex internal processing	2	1	2
Code must be reusable	1	1	1
Easy to install	5	0.5	2.5
Easy to use	5	0.5	2.5
Portable	1	2	2
Easy to change	3	1	3
Concurrent	5	1	5
Includes special security features	4	1	4
Provides direct access through third party	1	1	1
Special user training facilities are required	0	1	0
	Total weighted factor		35
	Technical factor		0.93

Total Weighted Factor = ∑(Extended Value)

TCF = 0.6+(0.01*Total Weighted Value)

Environment Factor	Description	Weight
F1	Familarity with Life-Cycle model used	1.5
F2	Application domain experience	0.5
F3	Experience with development methodologies used	1
F4	Analyst capability	0.5
F5	Team motivation	1
F6	Stability of requirements	2
F7	Use of part-time team members	-1
F8	Use of difficult programming language	-1

Environment Complexity Factor - ECF

- Each factor is assigned a value from 0 to 5

- Environment Total Factor is the sum of each factor weight multiplied by the perceived value

- Reflects the development team's experience

ECF = 1.4 + (-0.03*Environment Total Factor)

Assigning values to the Environmental Factors

F1 Familiar with Development Method: Score 0 if none of the team members are familiar with the development method to 5 if all the team has experience from using the method on several projects.

F2 Application experience: Score 0 if none of team members has experience with similar application domains to 5 if all the team has more than 5 years experience.

F3 Object-Oriented experience: Score 0 if the team is totally unfamiliar with OOAD to 5 if all the developers been trained in OOAD and have more than 5 years experience with it.

F4 Lead analyst capability: Score 0 if the lead analyst is a novice to 5 if he/she has more than 5 years experience from similar projects

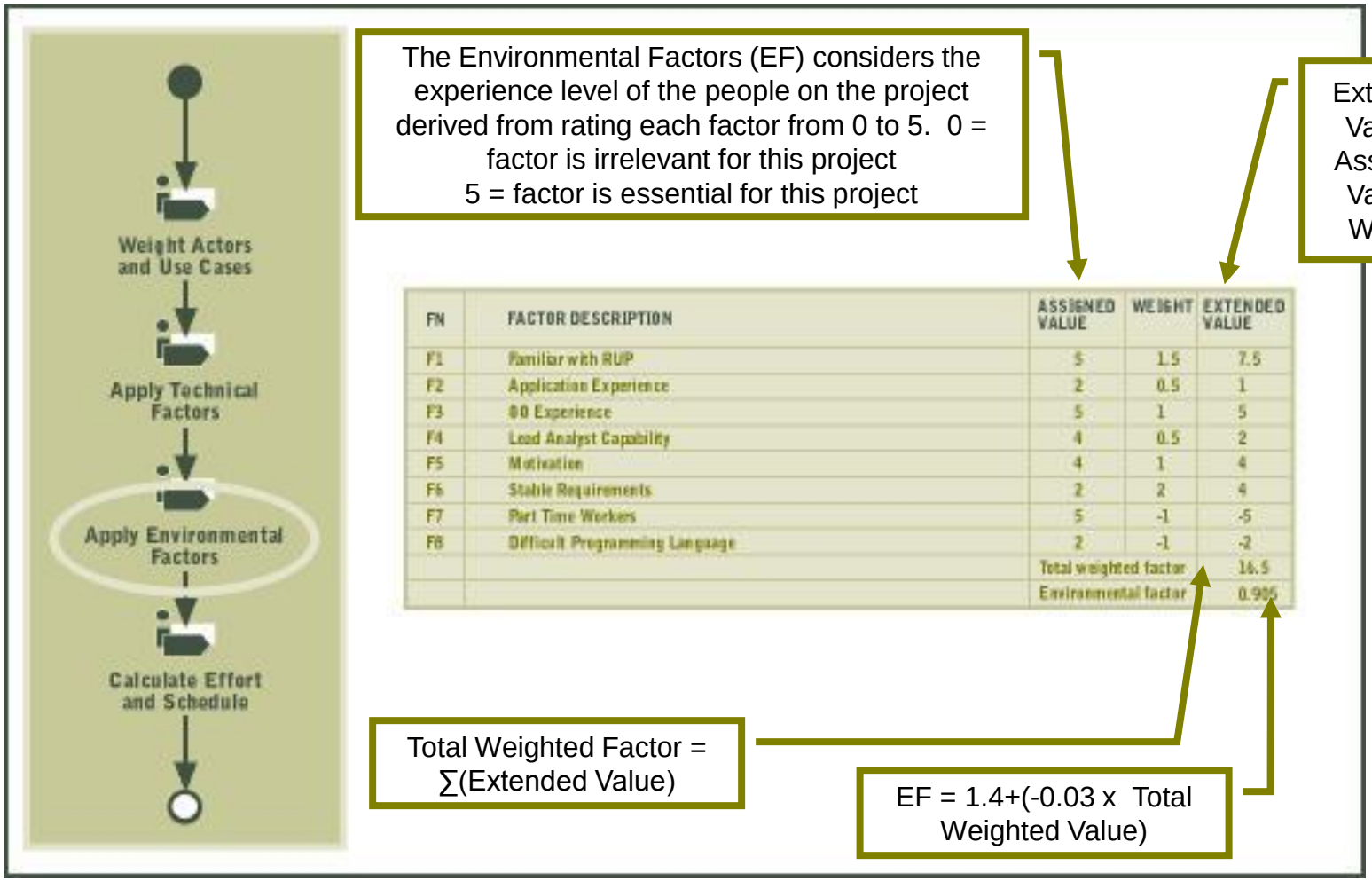
F5 Motivation: Score 0 if the project members lack motivation to 5 if the team is very motivated.

F6 Stable requirements: Score 0 if the requirements are likely to undergo constant changes to 5 if no changes in the requirements are expected.

F7 Part-time workers: Score 0 if there are no part-time workers to 5 if all are part-time workers.

F8 Experience with the tools: Score 0 if all the team members have been trained and have experience with the tools to 5 if they are all novices.

Apply Environmental Factors (ECF)



FN	FACTOR DESCRIPTION	ASSIGNED VALUE	WEIGHT	EXTENDED VALUE
F1	Familiar with RUP	5	1.5	7.5
F2	Application Experience	2	0.5	1
F3	OO Experience	5	1	5
F4	Lead Analyst Capability	4	0.5	2
F5	Motivation	4	1	4
F6	Stable Requirements	2	2	4
F7	Part Time Workers	5	-1	-5
F8	Difficult Programming Language	2	-1	-2
		Total weighted factor		16.5
		Environmental factor		0.906

Productivity Factor

- The adjusted UseCase points are multiplied with a productivity factor (hours per UseCase point).
- The value may range from 16 to 30 man-hour/UCP

Producing an Estimate

- The unadjusted actor weight, UAW, is calculated adding the weights for each actor.
- The unadjusted use case weights, UUCW, is calculated correspondingly.
- The unadjusted use case points, UUCP, = UAW + UUCW.
- The technical factor, TCF, = .6 + (.01*$\Sigma_{1..13} T_n$*$Weight_n$).
- The environmental factor, EF, = 1.4 + (-.03* $\Sigma_{1..8} F_n$*$Weight_n$).
- UCP = UUCP*TCF*EF
- Estimate = UCP * Productivity factor

Calculate Effort and Schedule

Use Case Points = (AW + UCW) x TCF x EF

Use Case Points = (8 + 35) x 0.93 x 0.905 = 36.19

USE CASE POINTS	36.19
PERSON HOUR FACTOR	28
TOTAL EFFORT	1013 PERSON HOURS

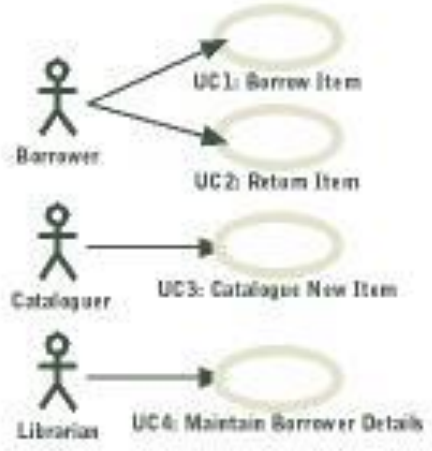
LOSS FACTOR	90%
ACTUAL EFFORT	1126 PERSON HOURS 141 PERSON DAYS
No. OF STAFF WORKING HOURS	2 8
SCHEDULE SCHEDULE	375.31 HOURS 47 DAYS

Improving Estimation Practices

- It is beneficial to combine estimates from different experts and estimation strategies.
  - The company's expert estimates are made bottom-up, the use case points method provides a top-down estimate.
- A supplementary use case based estimate provides a basis for adjusting the expert estimate.
- An estimation method may make more people competent to take part in estimation.

Summary

SCOPE



Use Cases provide a close approximation of project scope that provides an estimate that is extremely useful for high-level project planning.

+ takes into account technical and project risk

Based upon knowledge of the scope
Garbage IN Garbage OUT