Software Size and Effort Estimation

Dr. Aleš Živkovič, CISA, PRINCE 2

University of Maribor, Slovenia
Faculty of Electrical Engineering and Computer Science
e-mail: ales.zivkovic@uni-mb.si
http://www.feri.uni-mb.si/

Contents

Importance of Metrics

Maribor	**Linz**
Population: 119.000	189.000
Surface: 147 km2	96 km2
Elevation: 273 m	266 m

Faculty of EE & CS

- Staff - 255
  - 65 Professors
  - 70 Teaching Assistants
  - 50 Researchers
  - 37 Technicians
  - 25 Adminstration

- 2,000 undergraduate students
- 700 -800 new students every year
- 300 postgraduate students
- Add. 500 part-time students

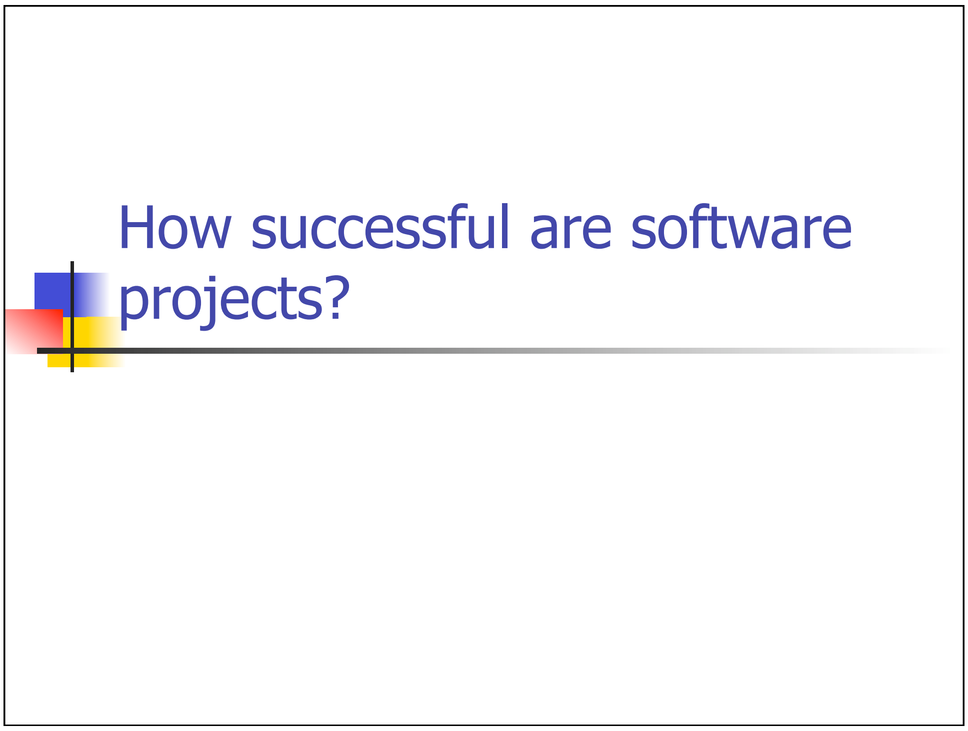
Estimating Software Size and Effort

When is the project successful?

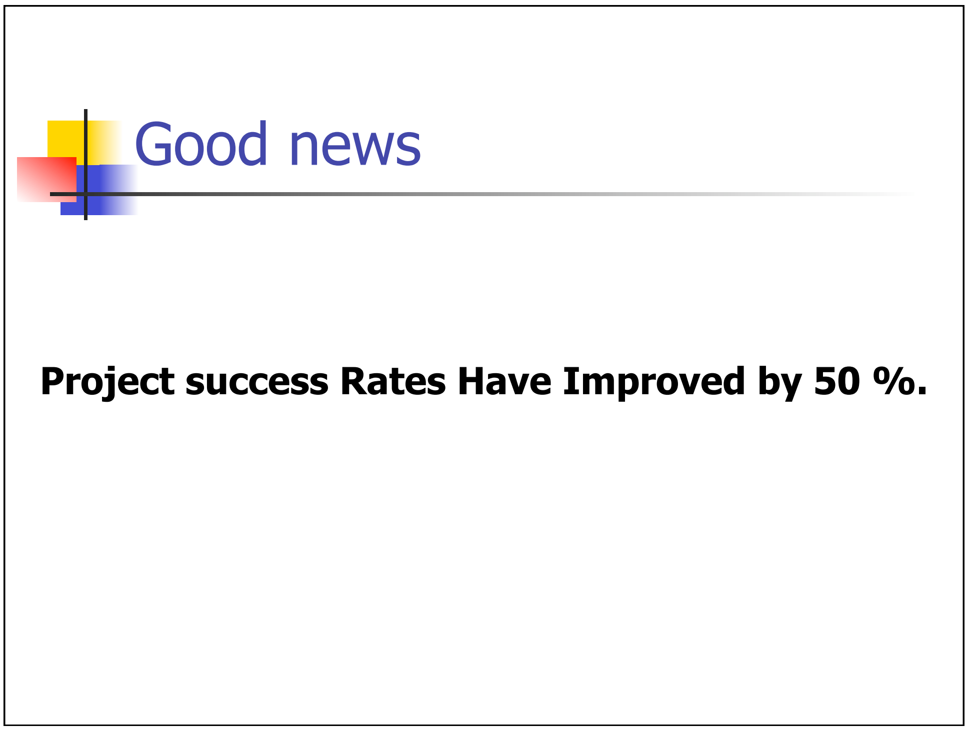
On-time)

On budget

Deliver all functions



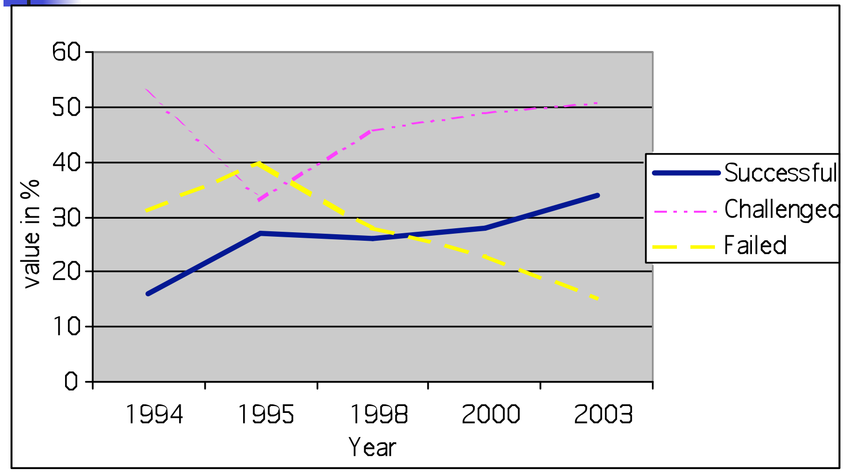
How successful are software projects?



Good news

Project success Rates Have Improved by 50 %.

Facts and Figures



Source: The Standish Group

And the reasons could be...

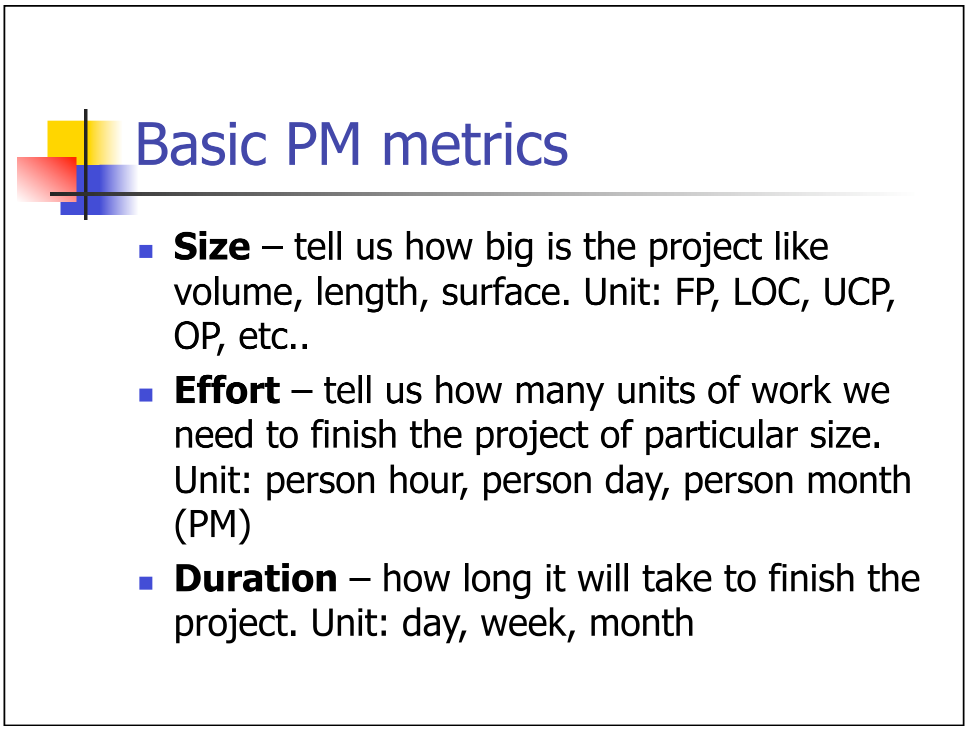
- Incomplete/Unstable Requirements
- Lack of User Involvement
- Unrealistic Expectations
- Lack of Executive Support
- Lack of Resources

...more likely reasons are

- software development process is not (partially) defined
- metrics are not introduced or are not in use
- coding without design
- low utilization of software development tools

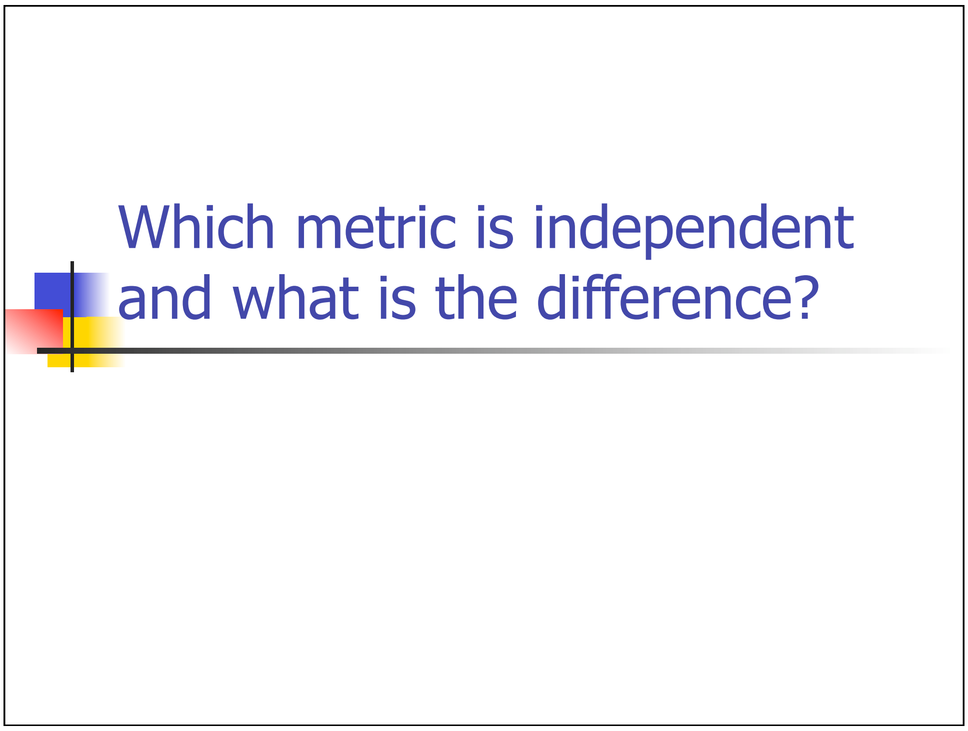
Basic project management questions

- How big is my project?
- How long it will take?
- How much it will cost?

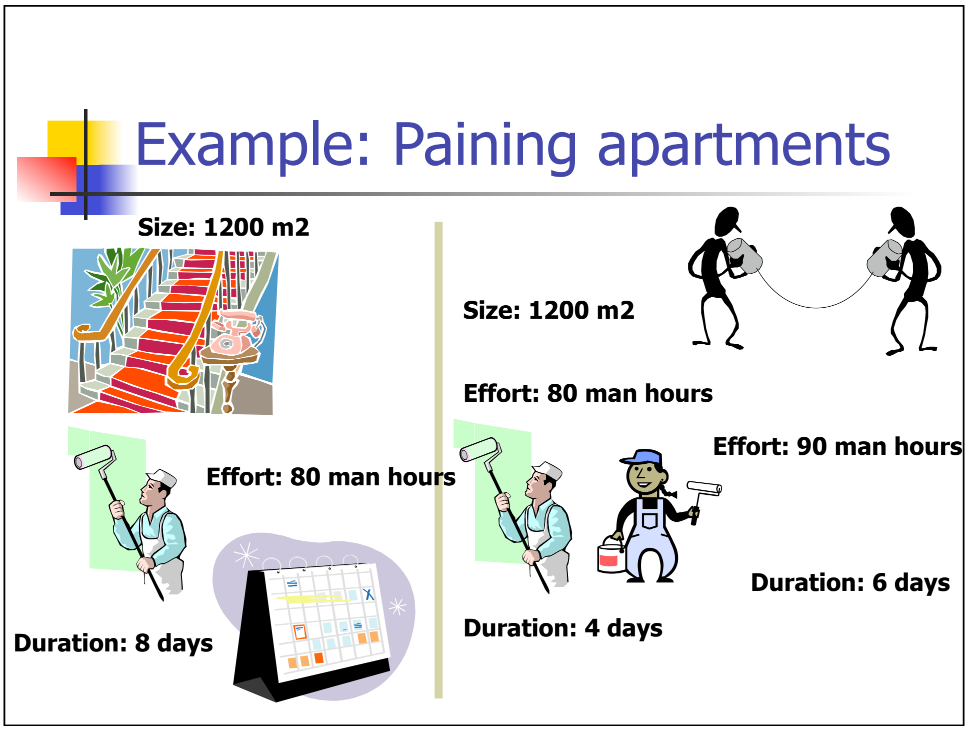


Basic PM metrics

- **Size** – tell us how big is the project like volume, length, surface. Unit: FP, LOC, UCP, OP, etc..
- **Effort** – tell us how many units of work we need to finish the project of particular size. Unit: person hour, person day, person month (PM)
- **Duration** – how long it will take to finish the project. Unit: day, week, month



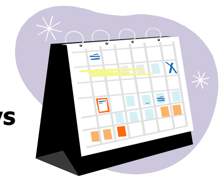
Which metric is independent and what is the difference?



Example: Paining apartments

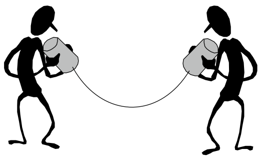
Size: 1200 m2

Effort: 80 man hours



Duration: 8 days

Size: 1200 m2

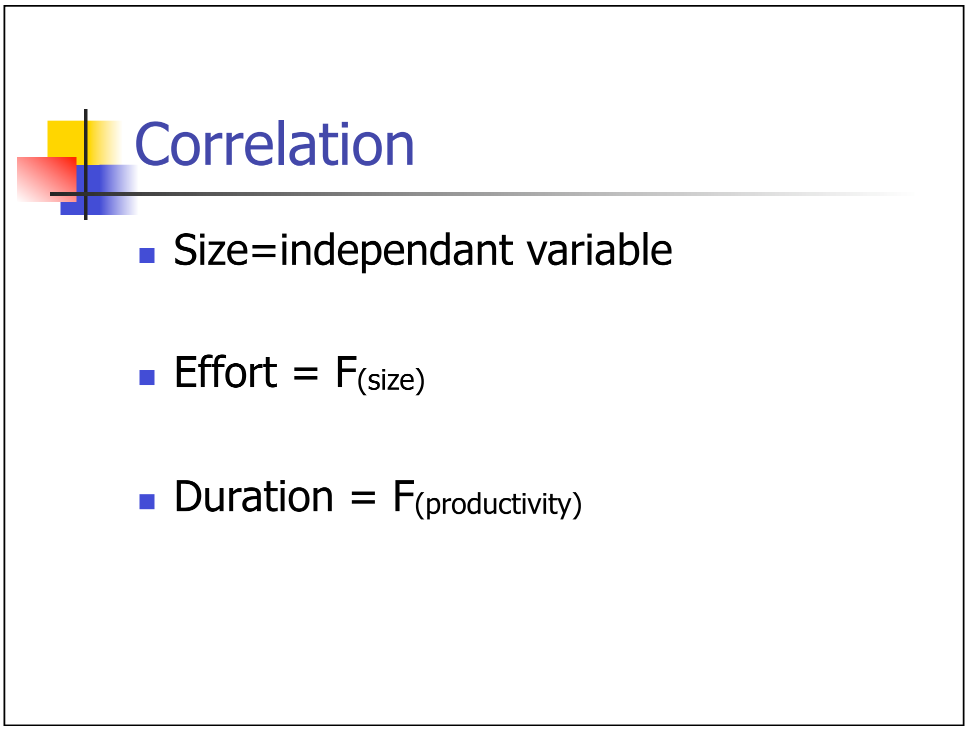


Effort: 80 man hours

Effort: 90 man hours

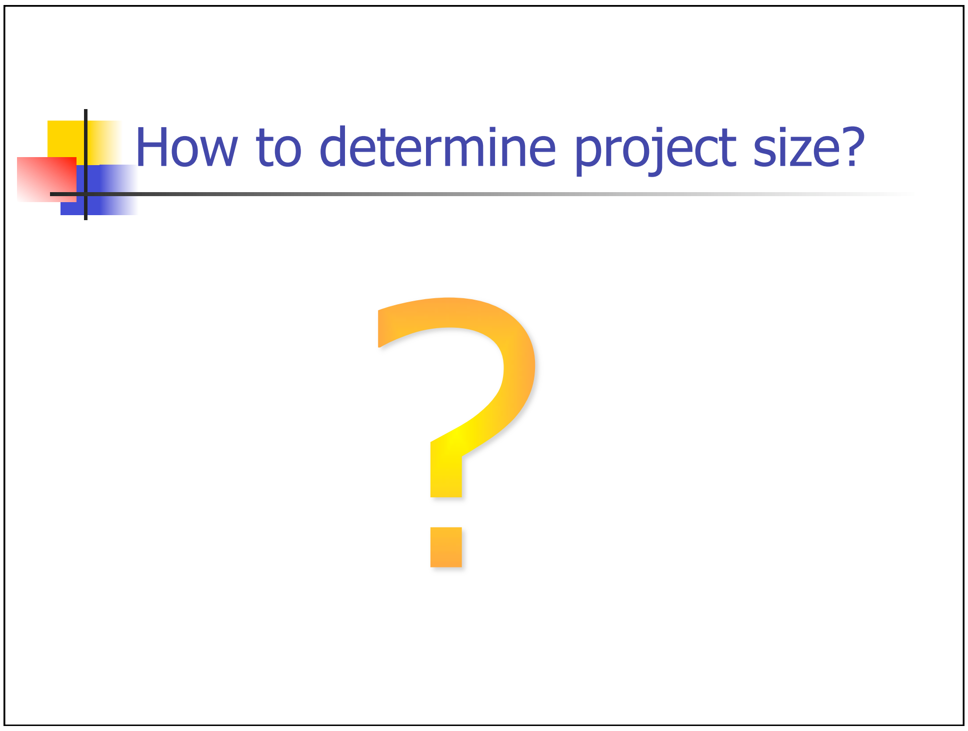
Duration: 4 days

Duration: 6 days



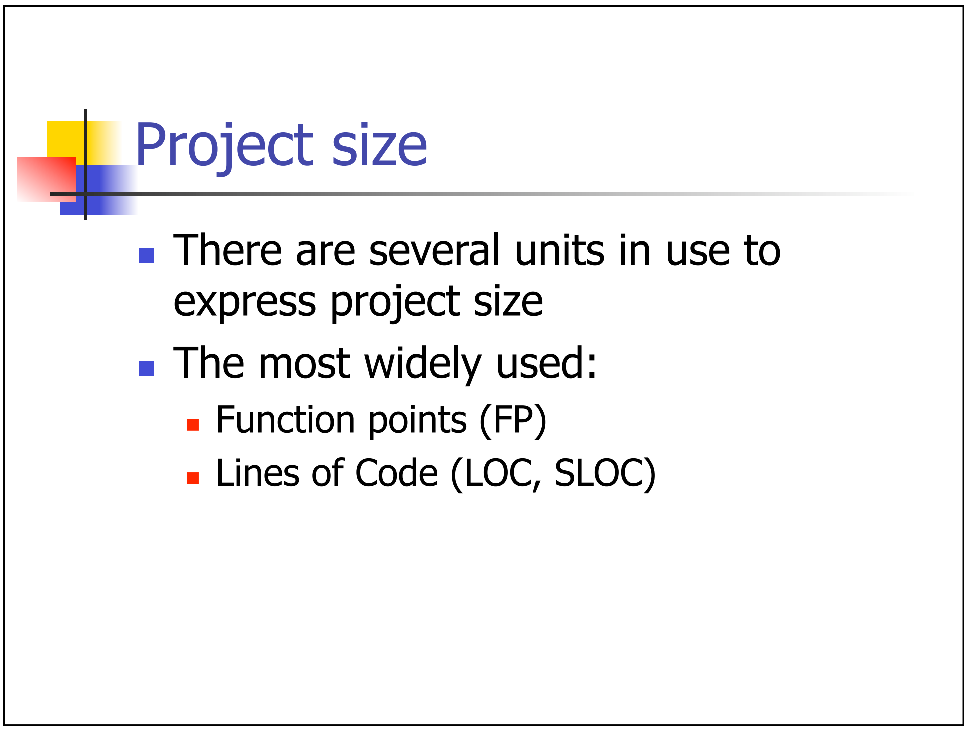
Correlation

- Size=independant variable
- Effort = F(size)
- Duration = F(productivity)



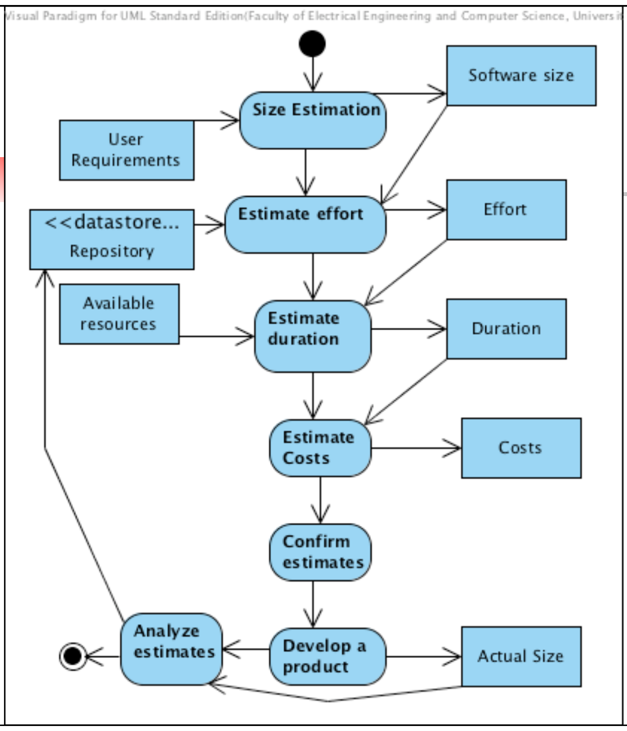
How to determine project size?

?



Project size

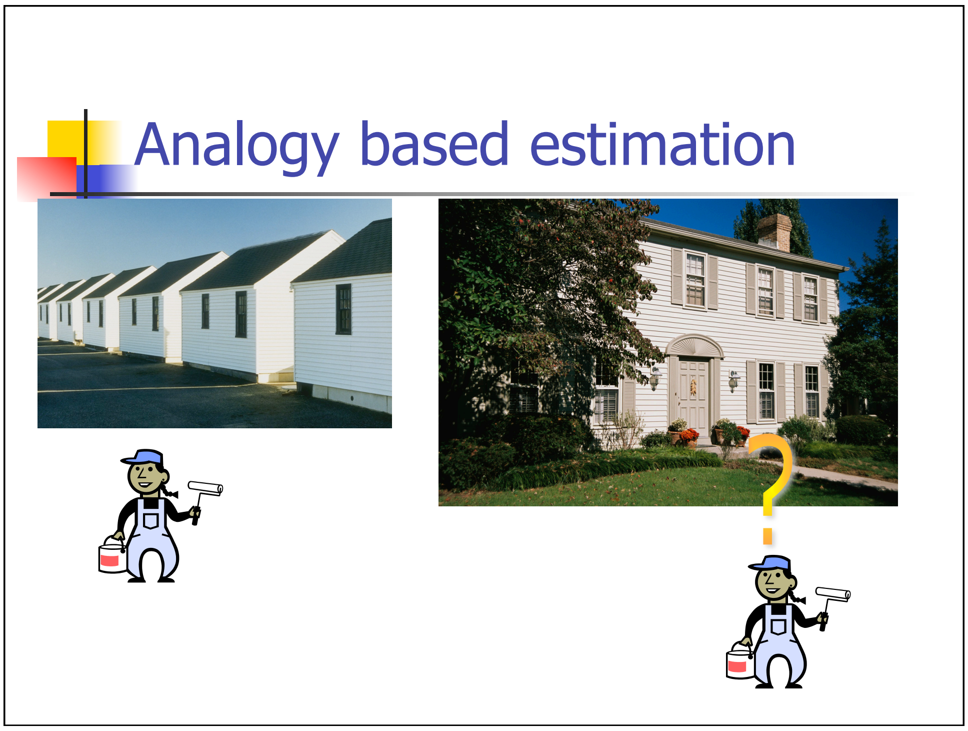
- There are several units in use to express project size
- The most widely used:
  - Function points (FP)
  - Lines of Code (LOC, SLOC)



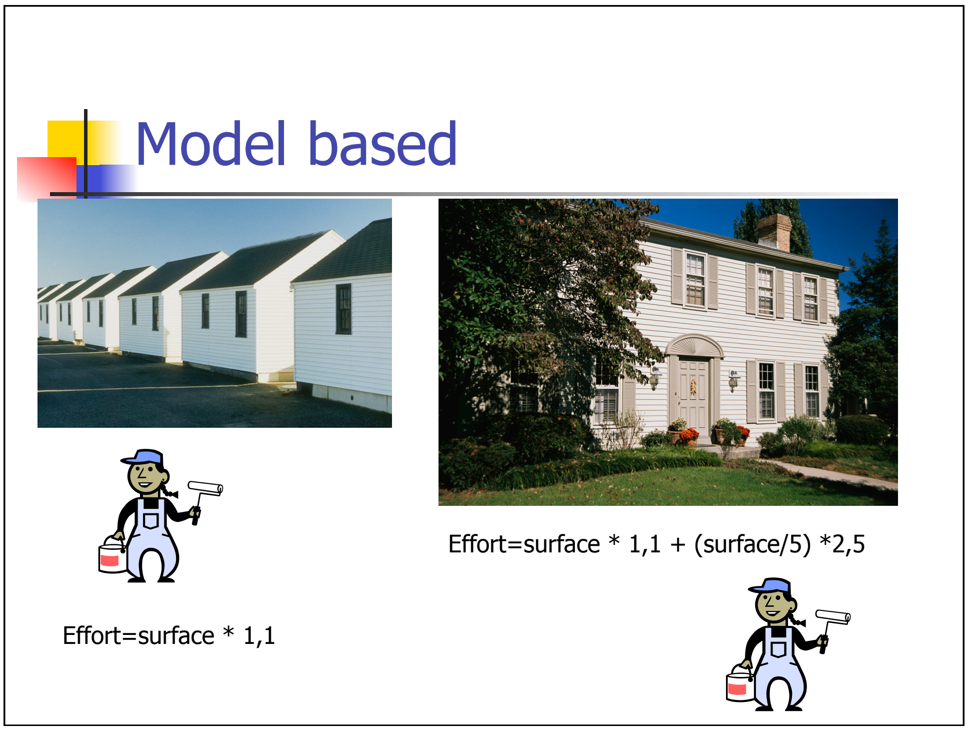
Estimation process

Size estimation

- **Analogy based**
  If we already did similar projects in the past and we have data (size, effort, duration) we can estimate the size of the project as the portion of the already finished project.
- **Model based approach**
  The approach is based on counting the properties of the product and the use of an algoritmic approach to calculate size (effort, duration). An example for this group are Function Points that transform the number of functions into product size.



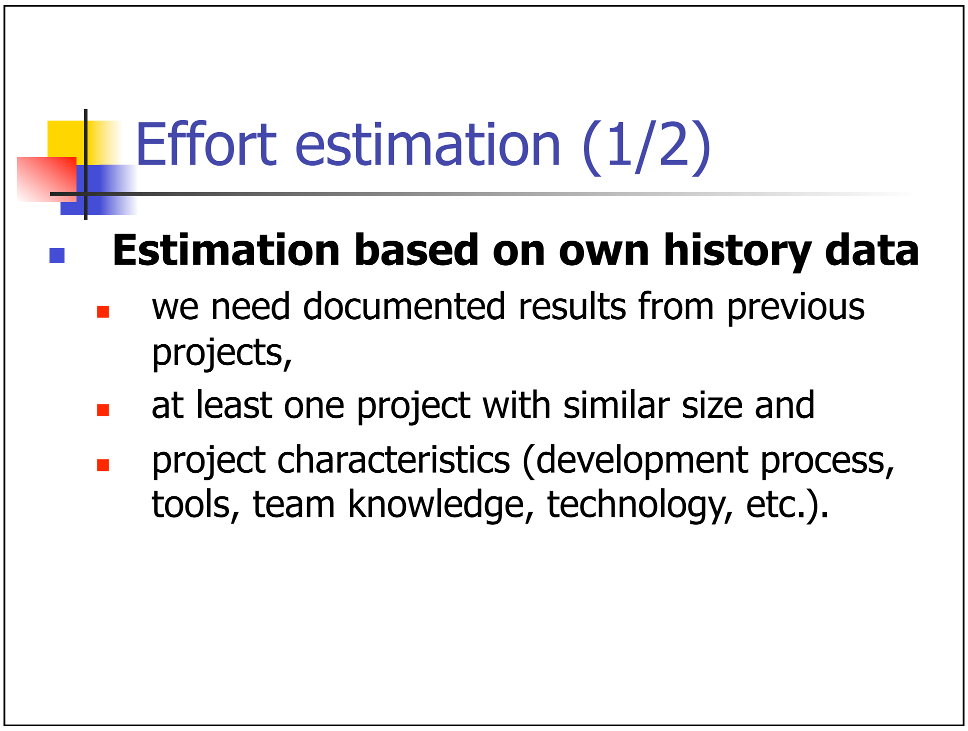
Analogy based estimation



Model based

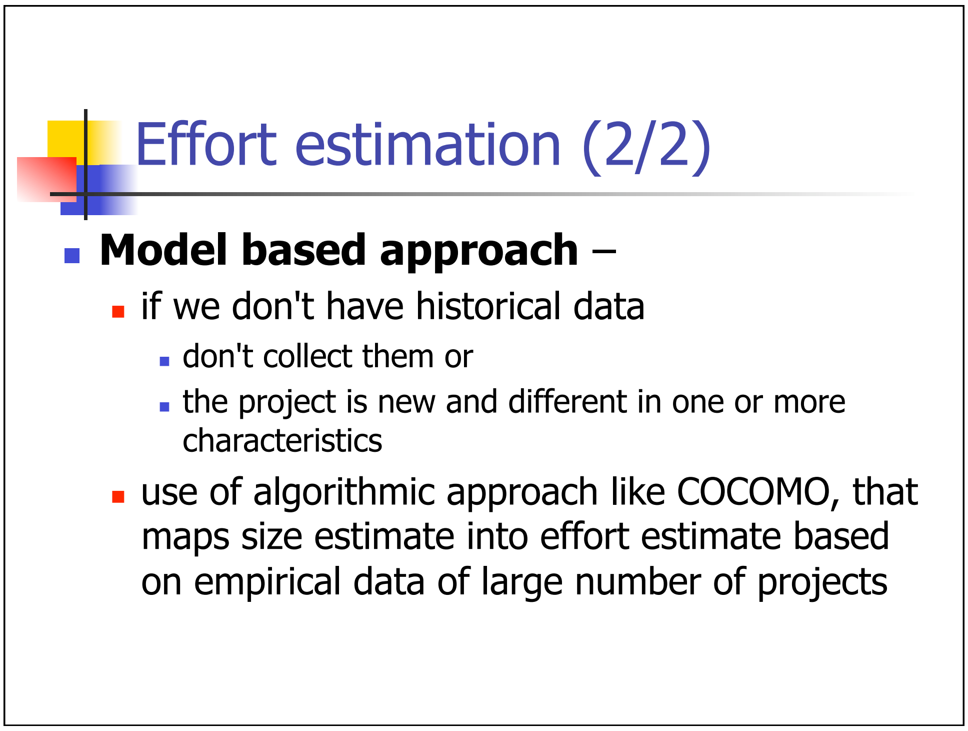
Effort=surface * 1,1 + (surface/5) *2,5

Effort=surface * 1,1



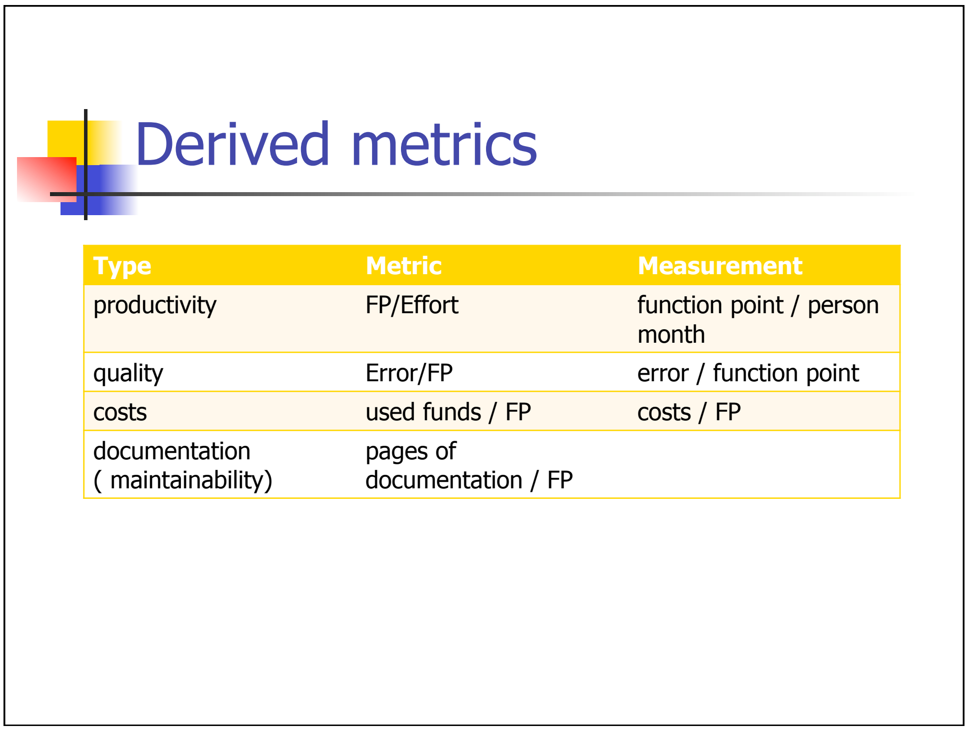
Effort estimation (1/2)

- **Estimation based on own history data**
  - we need documented results from previous projects,
  - at least one project with similar size and
  - project characteristics (development process, tools, team knowledge, technology, etc.).



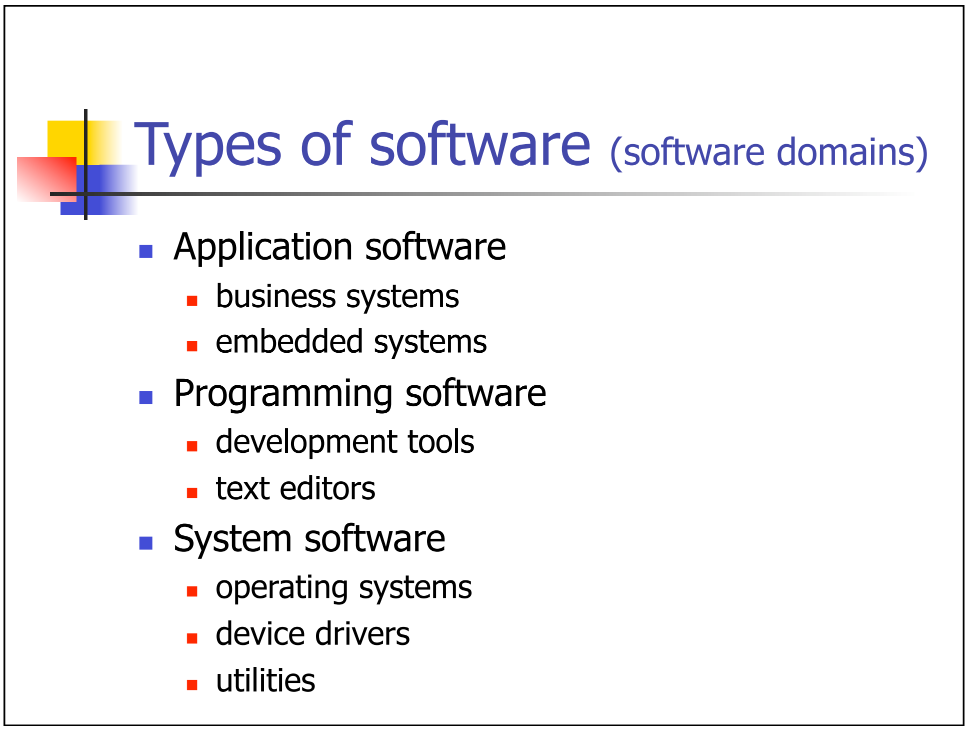
Effort estimation (2/2)

- **Model based approach** –
  - if we don't have historical data
    - don't collect them or
    - the project is new and different in one or more characteristics
  - use of algorithmic approach like COCOMO, that maps size estimate into effort estimate based on empirical data of large number of projects



Derived metrics

Type	Metric	Measurement
productivity	FP/Effort	function point / person month
quality	Error/FP	error / function point
costs	used funds / FP	costs / FP
documentation (maintainability)	pages of documentation / FP	

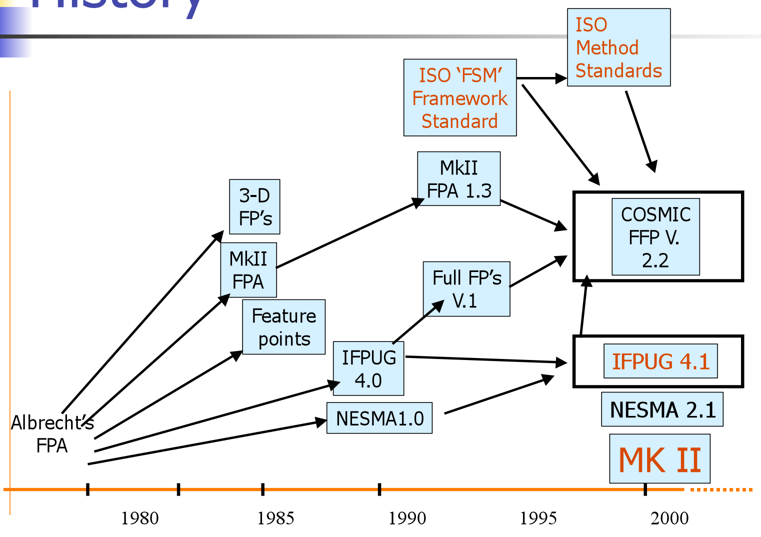


Types of software (software domains)

- Application software
  - business systems
  - embedded systems
- Programming software
  - development tools
  - text editors
- System software
  - operating systems
  - device drivers
  - utilities

FSM – Functional Size Measurement

History



Estimation methods overview

- **Function Point Analysis (FPA)**
  for application software (business domain) using structured development methods (non-OO)
- **Full Function Points (FFP) and Feature Points**
  focus on improving the original FPA method for real-time systems and system software as well as application software with complex algorithms (i.e. intense graphics, math calculations)
- **Mark II FPA –** introduces different abstraction model based on logical transactions and entities
- **NEtherlands Software Metrics users Association (NESMA) method** – similar to the original FPA method with improvements.

FPA method ISO/IEC 20926

- Function Point Analysis (FPA) metrics, was developed by Alan Albercht in 1979
- In 1984, the International Function Point Users Group (IFPUG) was set up to clarify rules, set standards, and promote their use and evolution

Feature Points

- Feature points are "superset" of FP
- The method adds a new software characteristics: “algorithms”
- Suitable for real-time, process-control and embedded software applications that have high algorithmic complexity

MK II FPA ISO/IEC 20968:2002

- Developed in the late 80’s by Charles Symons in the UK
- The main feature of the method is the simple measurement model
- There are only 3 components to consider:
  - **Inputs**: data coming into the software from the external environment (user)
  - **Outputs**: data going from the software to the user
  - **Entity References**: storage, retrieval and deletion of data from the permanent storage.

MK II FPA Calculation process

- The Functional Size (Function Point Index) is the weighted sum over all Logical Transactions
  - Input Data Element Types (Ni)
  - Data Entity Types Referenced (Ne)
  - Output Data Element Types (No)

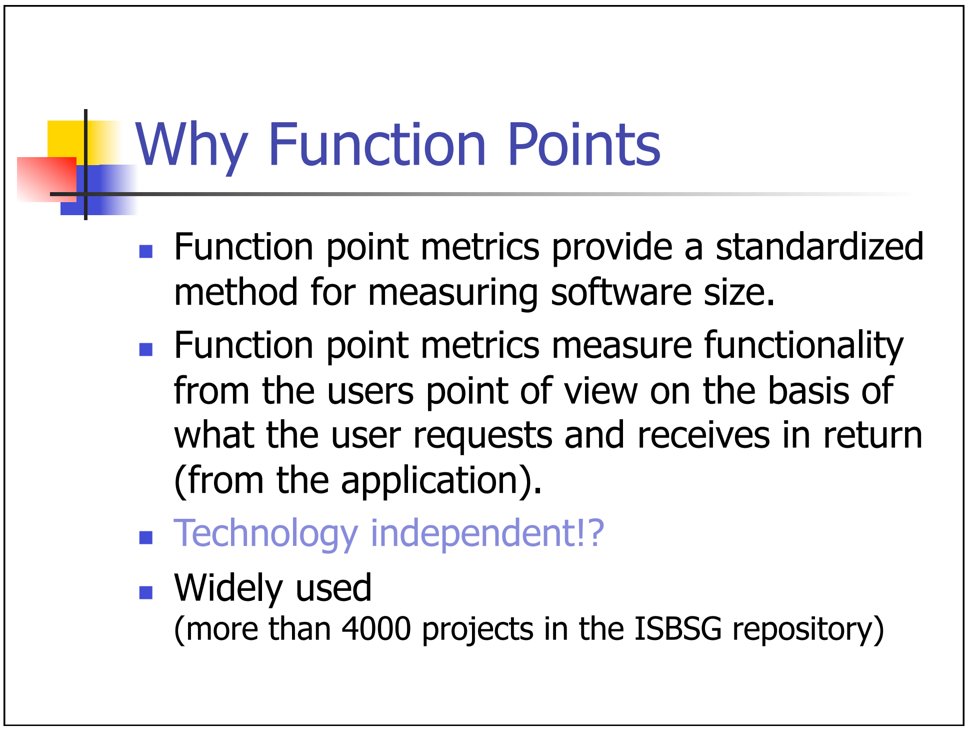
FPI = Wi * Sum(Ni) + We * Sum(Ne) + Wo * Sum(No)

Industry average weights are:

- Wi (Input Data Element Type) = 0.58
- We (Data Entity Type Reference) = 1.66
- Wo (Output Data Element Type)= 0.26

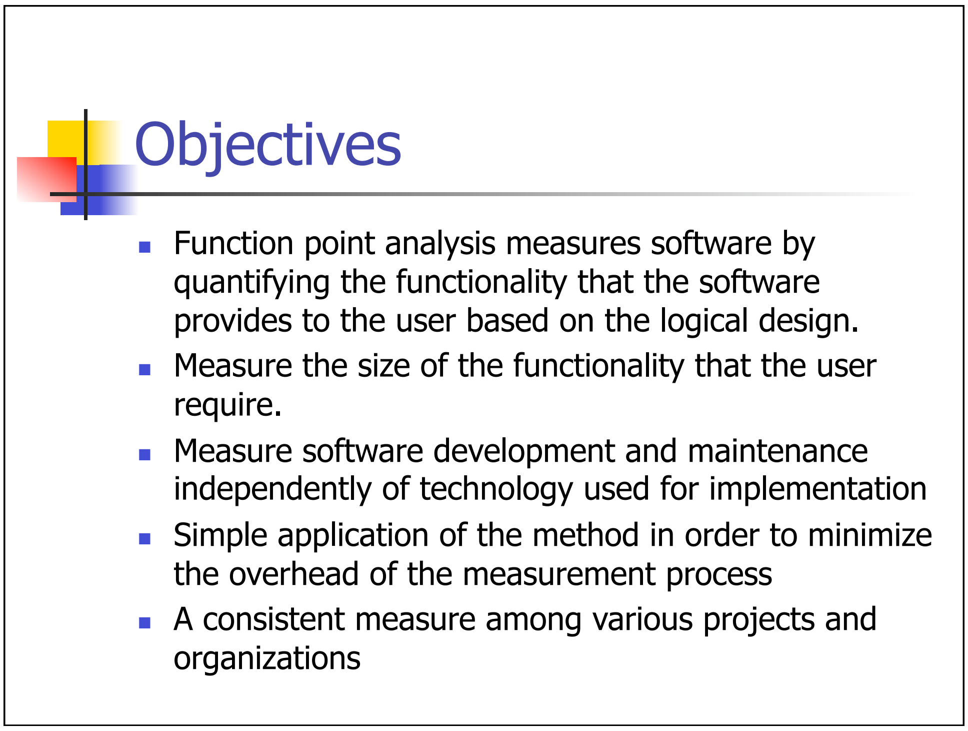
COSMIC FFP ISO/IEC 19761:2003

- Designed to measure the functional size in different domains (real-time, multi-layered software, process control and operating systems, business applications) using the same measurement scale.
- The method is compatible with modern specification methods such as UML and OO techniques.



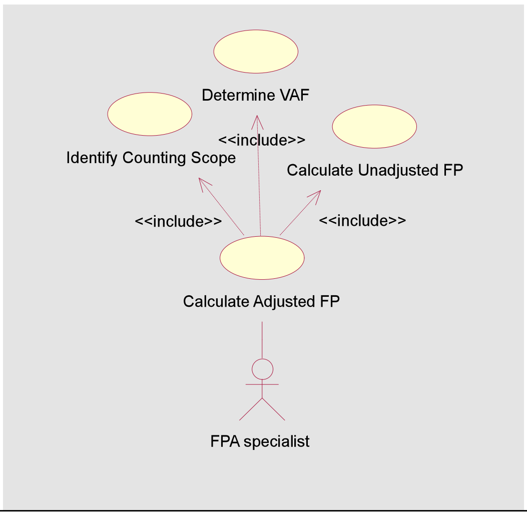
Why Function Points

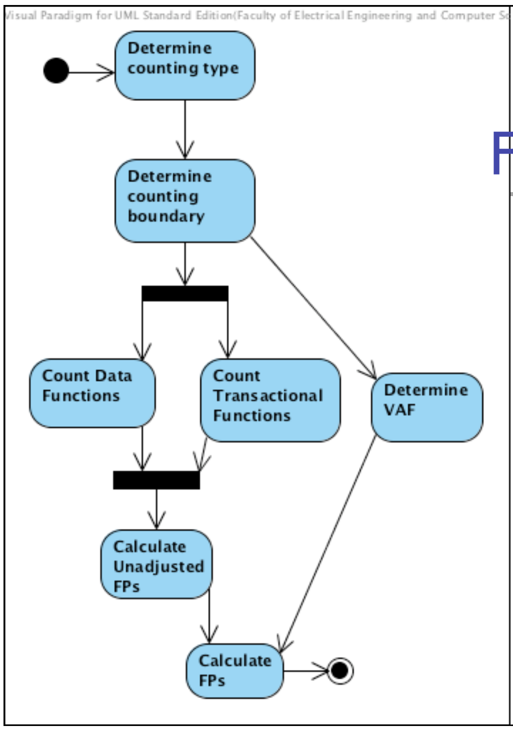
- Function point metrics provide a standardized method for measuring software size.
- Function point metrics measure functionality from the users point of view on the basis of what the user requests and receives in return (from the application).
- Technology independent!?
- Widely used
  (more than 4000 projects in the ISBSG repository)



Objectives

- Function point analysis measures software by quantifying the functionality that the software provides to the user based on the logical design.
- Measure the size of the functionality that the user require.
- Measure software development and maintenance independently of technology used for implementation
- Simple application of the method in order to minimize the overhead of the measurement process
- A consistent measure among various projects and organizations



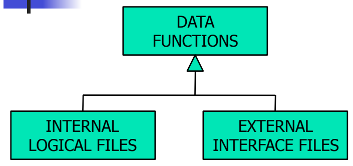

Visual Paradigm for UML Standard Edition(Faculty of Electrical Engineering and Computer Sc
Determine counting type
Determine counting boundary
Count Data Functions
Count Transactional Functions
Determine VAF
Calculate Unadjusted FPs
Calculate FPs

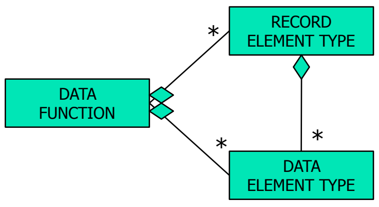
FPA method steps

Value Adjustment Factor:
VAF = 0.65 + 0.01 * GSC

Final size in FP:
FP = UFP *VAF

Data functions





$$NFT_{ILF}(N_{DET}, N_{RET}) = \begin{cases} 7; & ((1 \le N_{DET} \le 19) \wedge (1 \le N_{RET} \le 5)) \vee \\ & ((20 \le N_{DET} \le 50) \wedge (N_{RET} = 1)) \\ & ((1 \le N_{DET} \le 19) \wedge (6 \le N_{RET})) \vee \\ 10; & ((20 \le N_{DET} \le 50) \wedge (2 \le N_{RET} \le 5)) \vee \\ & ((51 \le N_{DET}) \wedge (N_{RET} = 1)) \\ 15; & ((20 \le N_{DET} \le 50) \wedge (6 \le N_{RET})) \vee \\ & ((51 \le N_{DET}) \wedge (2 \le N_{RET})) \end{cases}$$

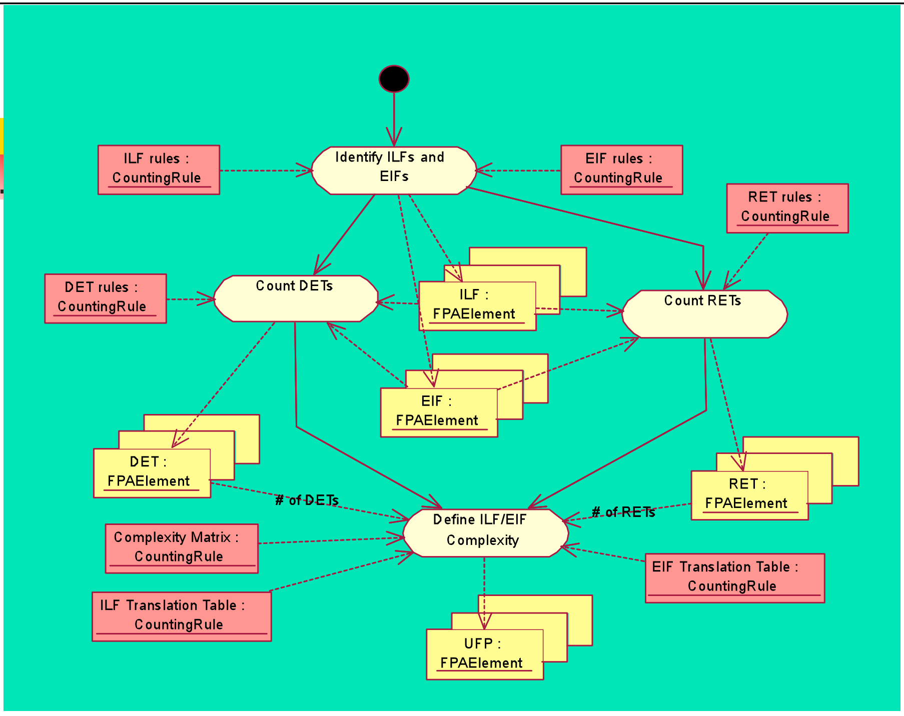
$$NFT_{EIF}(N_{DET}, N_{RET}) = \begin{cases} 5; & ((1 \le N_{DET} \le 19) \wedge (1 \le N_{RET} \le 5)) \vee \\ & ((20 \le N_{DET} \le 50) \wedge (N_{RET} = 1)) \\ & ((1 \le N_{DET} \le 19) \wedge (6 \le N_{RET})) \vee \\ 7; & ((20 \le N_{DET} \le 50) \wedge (2 \le N_{RET} \le 5)) \vee \\ & ((51 \le N_{DET}) \wedge (N_{RET} = 1)) \\ 10; & ((20 \le N_{DET} \le 50) \wedge (6 \le N_{RET})) \vee \\ & ((51 \le N_{DET}) \wedge (2 \le N_{RET})) \end{cases}$$

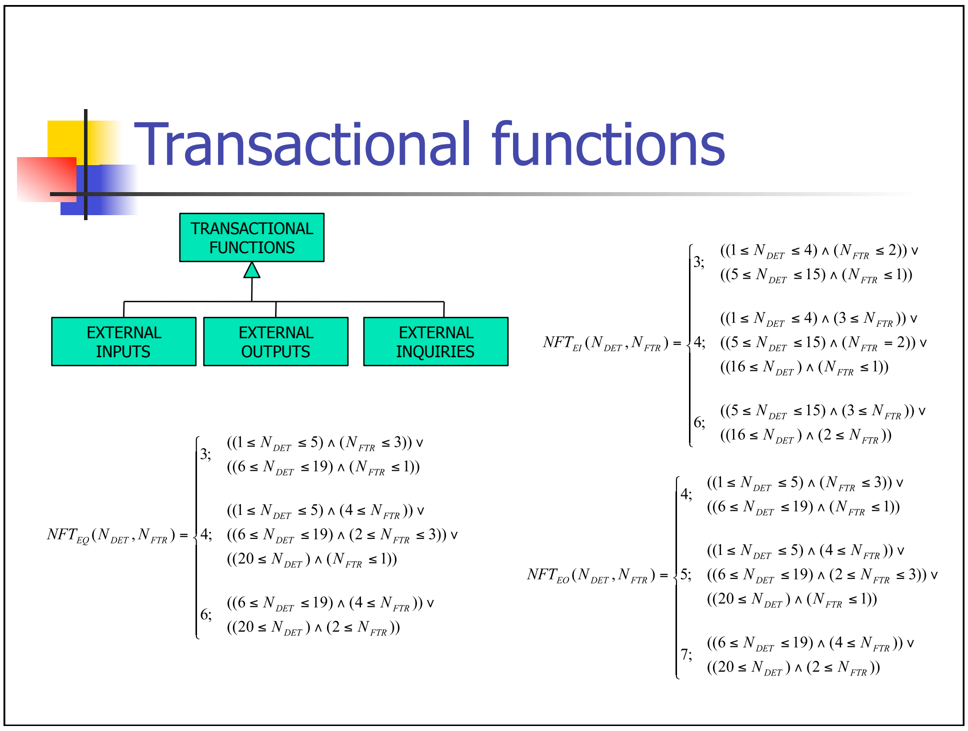
Data Types Definitions

- **Internal Logical File** (ILF): a user identifiable group of logically related data or control information maintained by the application (the count is done)
- **External Interface File** (EIF): a user identifiable group of logically related data or control information referenced by the application, but maintained by another application.
- The EIF in one application is ILF in another application – depends on the application boundary and consequently the counting boundary.

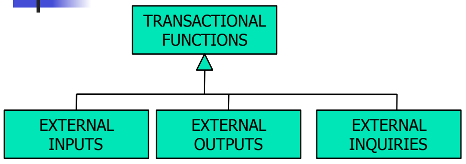
Complexity table for data functions

	1 to 19 DET	20 to 50 DET	51 or more DET
1 RET	Low	Low	Average
2 to 5 RET	Low	Average	High
6 or more RET	Average	High	High





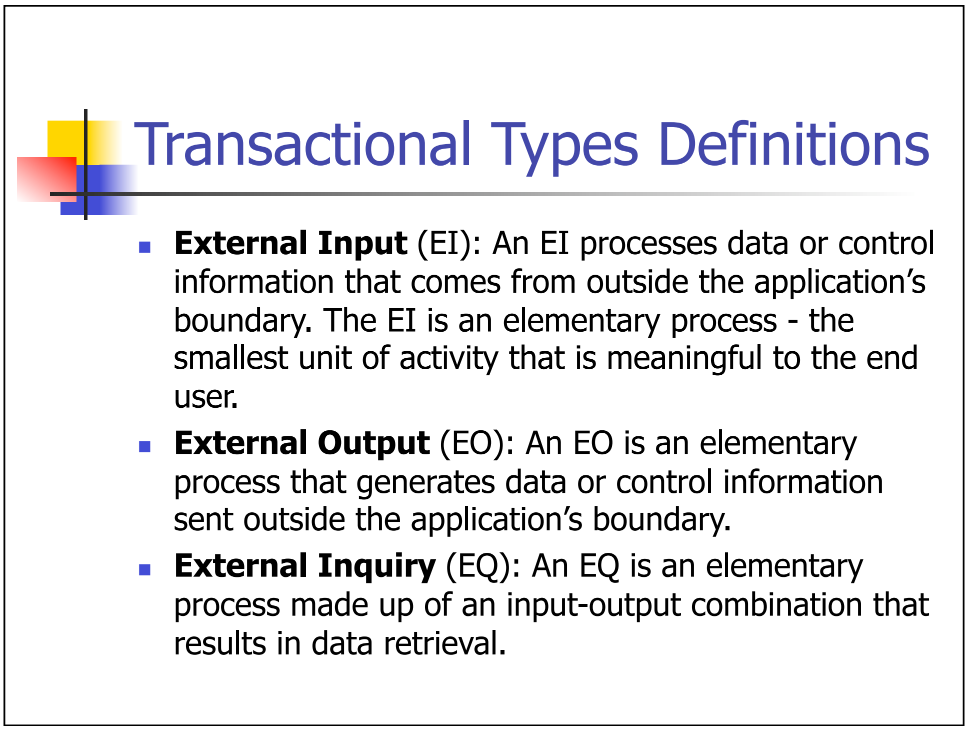
Transactional functions



$$NFT_{EI}(N_{DET}, N_{FTR}) = \begin{cases} 3; & ((1 \le N_{DET} \le 4) \wedge (N_{FTR} \le 2)) \vee \\ & ((5 \le N_{DET} \le 15) \wedge (N_{FTR} \le 1)) \\ & ((1 \le N_{DET} \le 4) \wedge (3 \le N_{FTR})) \vee \\ 4; & ((5 \le N_{DET} \le 15) \wedge (N_{FTR} = 2)) \vee \\ & ((16 \le N_{DET}) \wedge (N_{FTR} \le 1)) \\ 6; & ((5 \le N_{DET} \le 15) \wedge (3 \le N_{FTR})) \vee \\ & ((16 \le N_{DET}) \wedge (2 \le N_{FTR})) \end{cases}$$

$$NFT_{EQ}(N_{DET}, N_{FTR}) = \begin{cases} 3; & ((1 \le N_{DET} \le 5) \wedge (N_{FTR} \le 3)) \vee \\ & ((6 \le N_{DET} \le 19) \wedge (N_{FTR} \le 1)) \\ & ((1 \le N_{DET} \le 5) \wedge (4 \le N_{FTR})) \vee \\ 4; & ((6 \le N_{DET} \le 19) \wedge (2 \le N_{FTR} \le 3)) \vee \\ & ((20 \le N_{DET}) \wedge (N_{FTR} \le 1)) \\ 6; & ((6 \le N_{DET} \le 19) \wedge (4 \le N_{FTR})) \vee \\ & ((20 \le N_{DET}) \wedge (2 \le N_{FTR})) \end{cases}$$

$$NFT_{EO}(N_{DET}, N_{FTR}) = \begin{cases} 4; & ((1 \le N_{DET} \le 5) \wedge (N_{FTR} \le 3)) \vee \\ & ((6 \le N_{DET} \le 19) \wedge (N_{FTR} \le 1)) \\ & ((1 \le N_{DET} \le 5) \wedge (4 \le N_{FTR})) \vee \\ 5; & ((6 \le N_{DET} \le 19) \wedge (2 \le N_{FTR} \le 3)) \vee \\ & ((20 \le N_{DET}) \wedge (N_{FTR} \le 1)) \\ 7; & ((6 \le N_{DET} \le 19) \wedge (4 \le N_{FTR})) \vee \\ & ((20 \le N_{DET}) \wedge (2 \le N_{FTR})) \end{cases}$$



Transactional Types Definitions

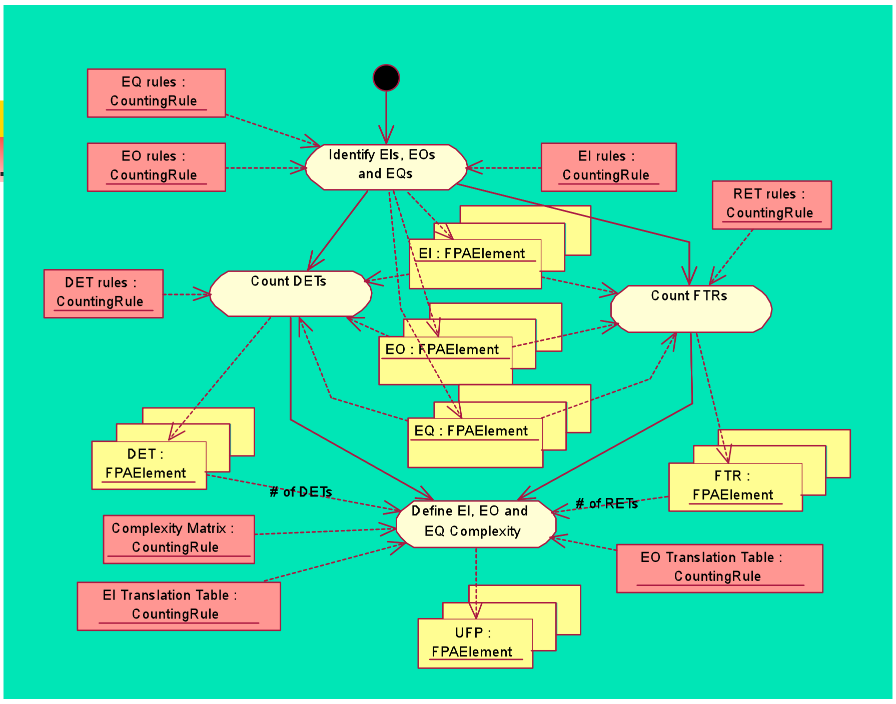
- **External Input** (EI): An EI processes data or control information that comes from outside the application’s boundary. The EI is an elementary process - the smallest unit of activity that is meaningful to the end user.
- **External Output** (EO): An EO is an elementary process that generates data or control information sent outside the application’s boundary.
- **External Inquiry** (EQ): An EQ is an elementary process made up of an input-output combination that results in data retrieval.

Complexity table for EO and EQ

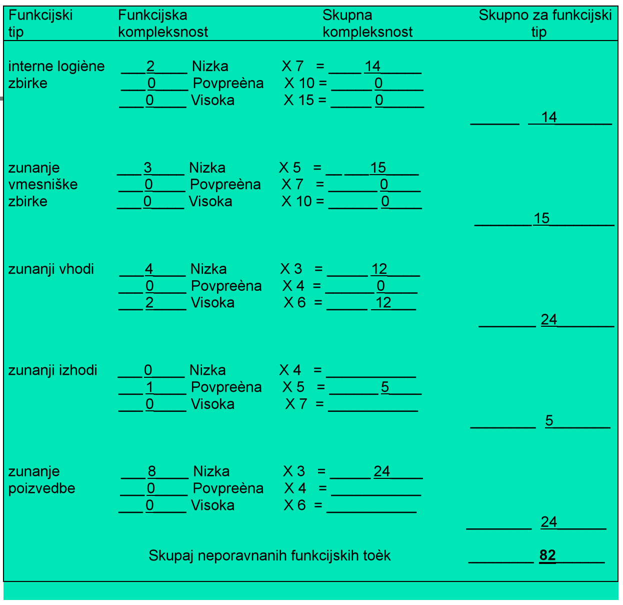
	1 to 5 DET	6 to 19 DET	20 or more DET
0 to 1 FTR	Low	Low	Average
2 to 3 FTRs	Low	Average	High
4 or more FTRs	Average	High	High

Complexity table for EI

	1 to 4 DET	5 to 15 DET	16 or more DET
0 to 1 FTR	Low	Low	Average
2 FTRs	Low	Average	High
3 or more FTRs	Average	High	High



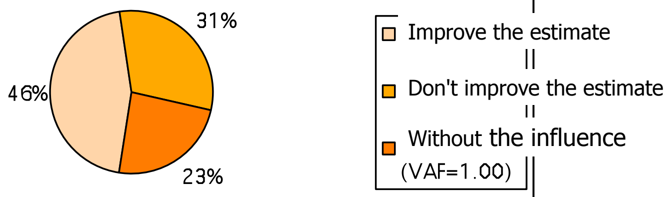
Manual Count (table in Slovene)



Funkcijski tip	Funkcijska kompleksnost		Skupna kompleksnost	Skupno za funkcijski tip
interne logiène zbirke	2	Nizka	X 7 = 14	
	0	Povpreèna	X 10 = 0	
	0	Visoka	X 15 = 0	14
zunanje vmesniške zbirke	3	Nizka	X 5 = 15	
	0	Povpreèna	X 7 = 0	
	0	Visoka	X 10 = 0	15
zunanji vhodi	4	Nizka	X 3 = 12	
	0	Povpreèna	X 4 = 0	
	2	Visoka	X 6 = 12	24
zunanji izhodi	0	Nizka	X 4 =	
	1	Povpreèna	X 5 = 5	
	0	Visoka	X 7 =	5
zunanje poizvedbe	8	Nizka	X 3 = 24	
	0	Povpreèna	X 4 =	
	0	Visoka	X 6 =	24
		Skupaj neporavnanih funkcijskih toèk		**82**

VAF influence to the final count

The influence of the Value Adjusted Factor to the size estimate (empirical analysis)



From Size to Effort 1/2

- To calculate Effort using software size you need a model i.e. formula that transforms size into effort. The formula could be:
  - based on empirical data
  - based on mathematical model
- Before using a formula verify if it is valid for your project!!!

From Size to Effort

$$E = n * (Size)^m$$ GENERAL

- COCOMO II

$$E = 2,4 * (KLOC)^{1,05}$$ for organic model

- Using history data and/or repository

$$Effort = 209,9 * Size^{0,155} * TeamSize^{1,398}$$ Size in FPs

Source: ISBSG repository valid for PC and 3GL development

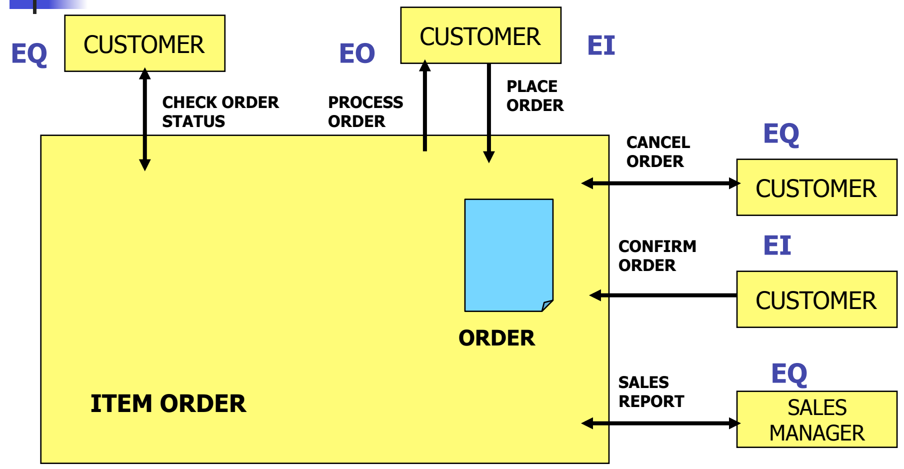
Function Points

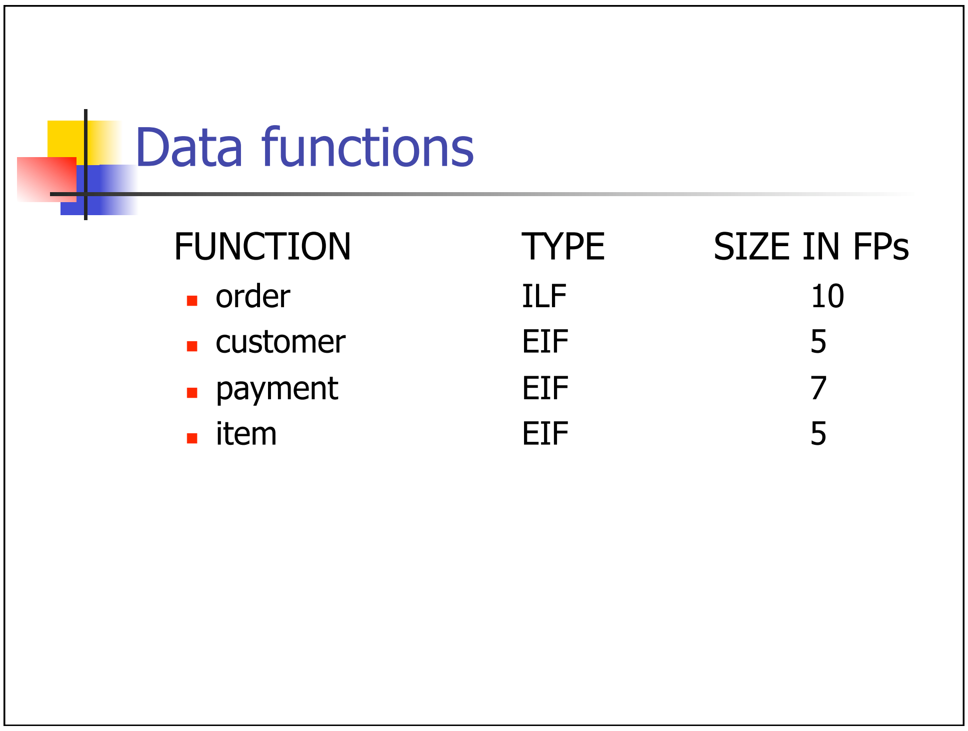
Example

Web store

- Functions
  - place order (17 atributes)
  - check order status (17 atributes)
  - process order (12 atributes)
  - cancel order (17 atributes)
  - confirm order (15 atributes)
  - sales report (6 atributes)

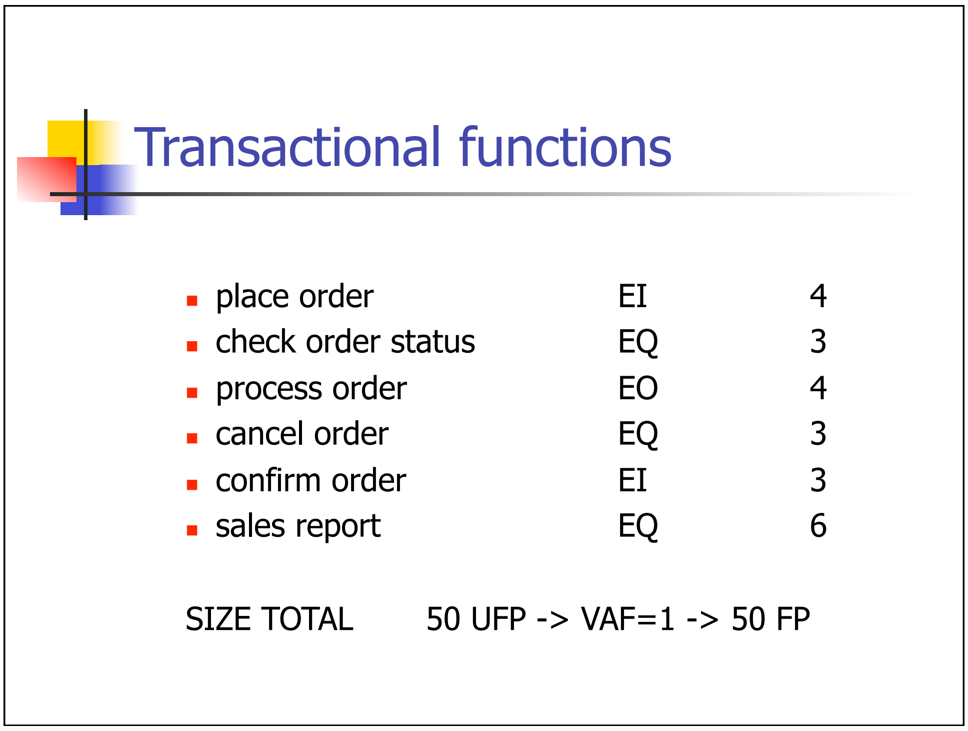
Determine element type





Data functions

FUNCTION	TYPE	SIZE IN FPs
▪ order	ILF	10
▪ customer	EIF	5
▪ payment	EIF	7
▪ item	EIF	5

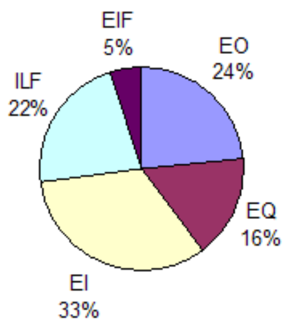


Transactional functions

▪ place order	EI	4
▪ check order status	EQ	3
▪ process order	EO	4
▪ cancel order	EQ	3
▪ confirm order	EI	3
▪ sales report	EQ	6

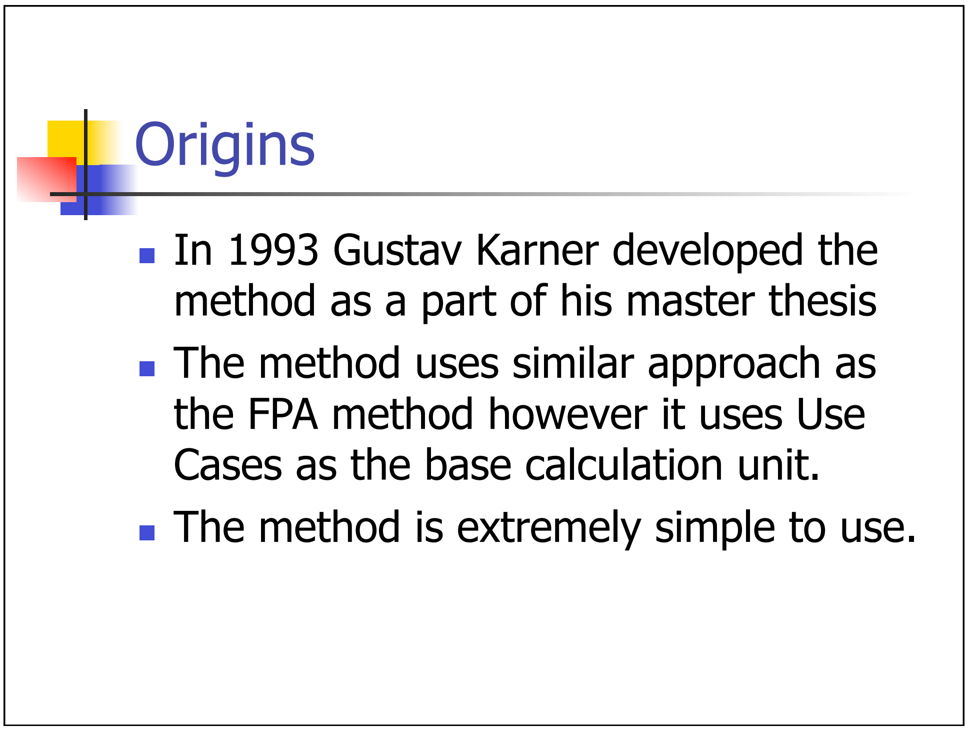
SIZE TOTAL 50 UFP -> VAF=1 -> 50 FP

Calculation with partial data



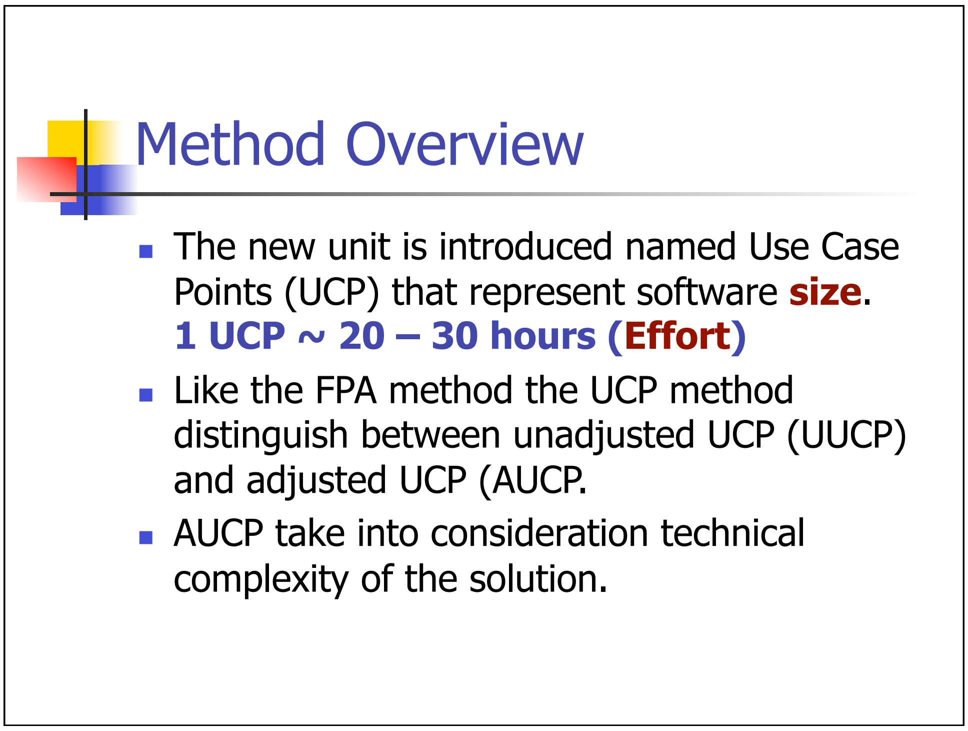
Use Case Points

Method for estimating software size in object-oriented development



Origins

- In 1993 Gustav Karner developed the method as a part of his master thesis
- The method uses similar approach as the FPA method however it uses Use Cases as the base calculation unit.
- The method is extremely simple to use.



Method Overview

- The new unit is introduced named Use Case Points (UCP) that represent software **size**.
  1 UCP ~ 20 – 30 hours (Effort)
- Like the FPA method the UCP method distinguish between unadjusted UCP (UUCP) and adjusted UCP (AUCP.
- AUCP take into consideration technical complexity of the solution.

Formulas

$$AUCP = UUCP * TCF * EF$$

$$UUCP = UAW + UUCW$$

$$TCF = 0.6 + (0.01 * TFactor)$$

$$EF = 1.4 + (-0.03 * EFactor)$$

$$TFactor = \sum_i T_i * W(T)_i$$

$$EFactor = \sum_i E_i * W(E)_i$$

Explanation of the terms

- **UAW** (Unadjusted actor weight) – how complex is the actor's role, weight between 1 and 3
- **UUCW** (Unadjusted UC weight) – complexity of the use case, weight between 1 and 5
- **TFactor** – sum of individual technical factors, weight between 0 and 5
- **EFactor** – sum of individual environment factors, weight between 0 and 5

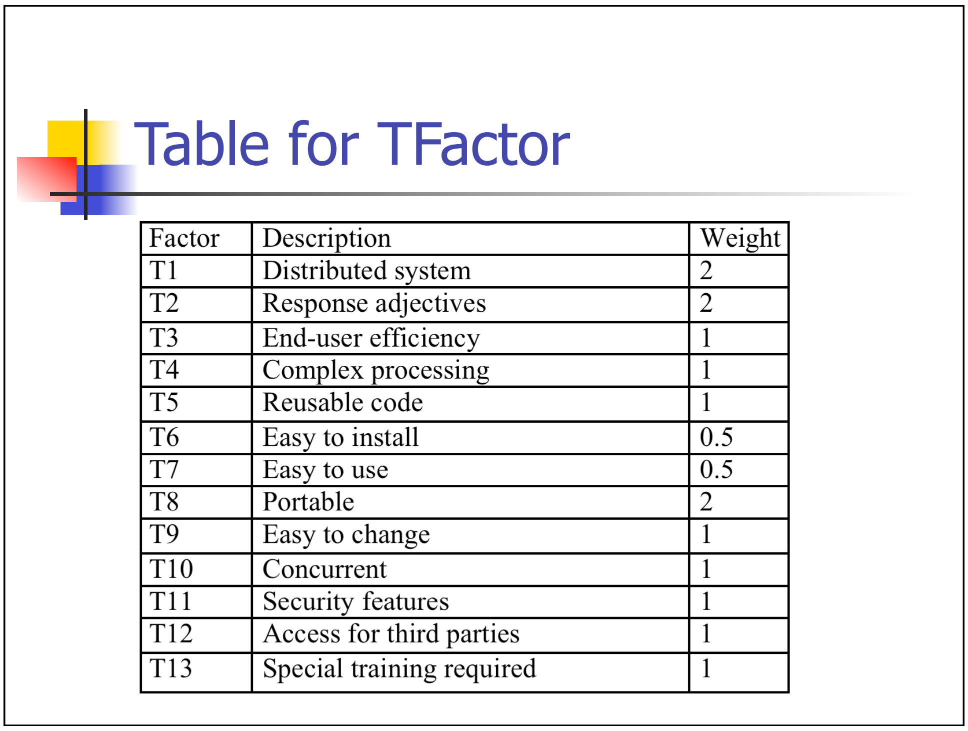


Table for TFactor

Factor	Description	Weight
T1	Distributed system	2
T2	Response adjectives	2
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	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	Security features	1
T12	Access for third parties	1
T13	Special training required	1

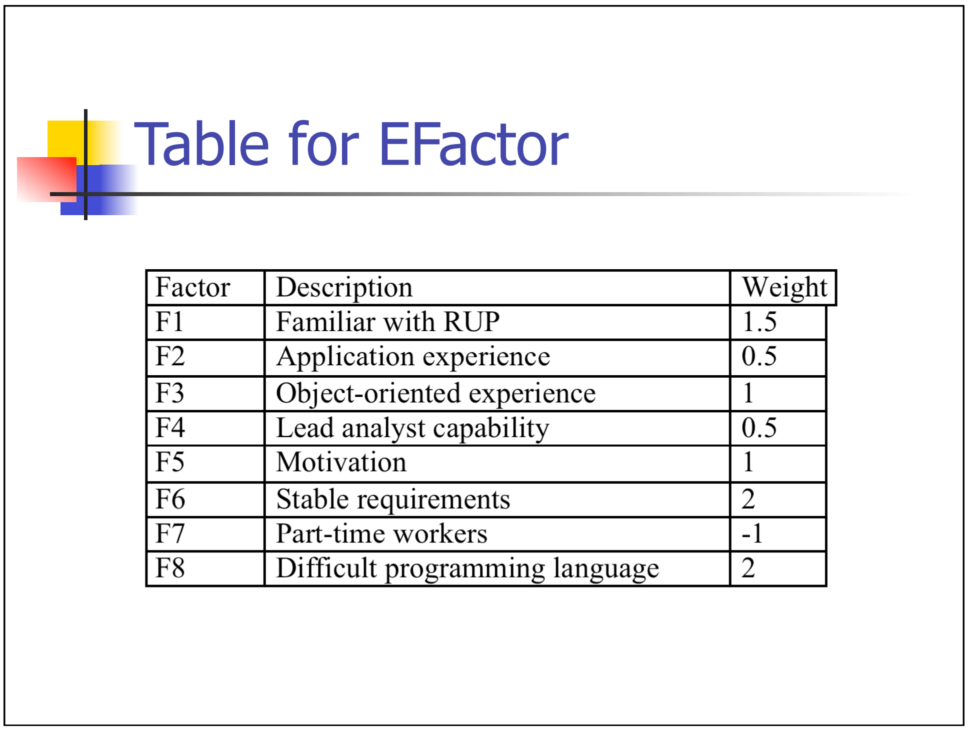


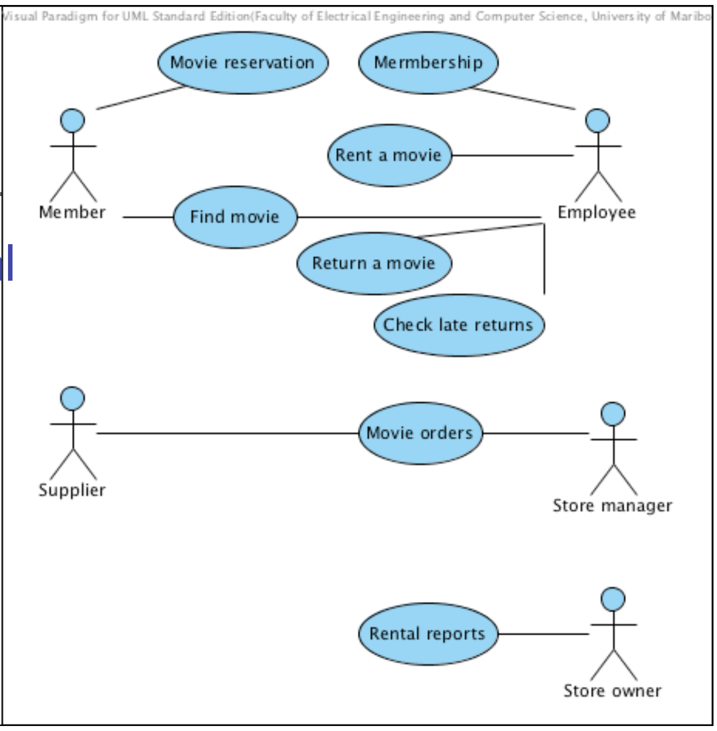
Table for EFactor

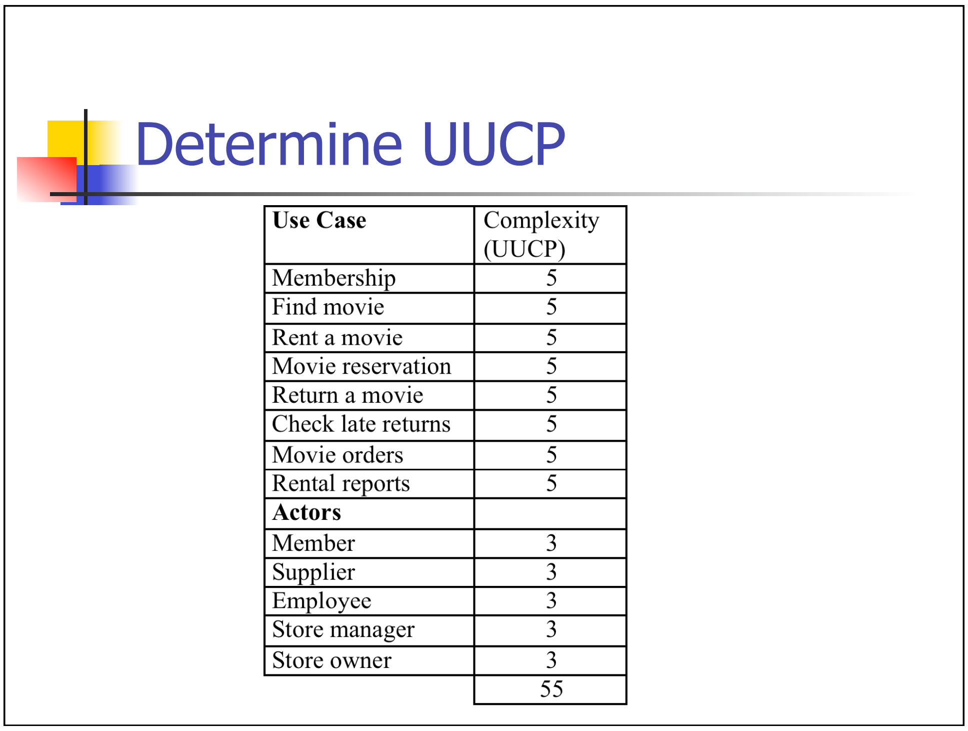
Factor	Description	Weight
F1	Familiar with RUP	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2
F7	Part-time workers	-1
F8	Difficult programming language	2

Use Case Points

Example

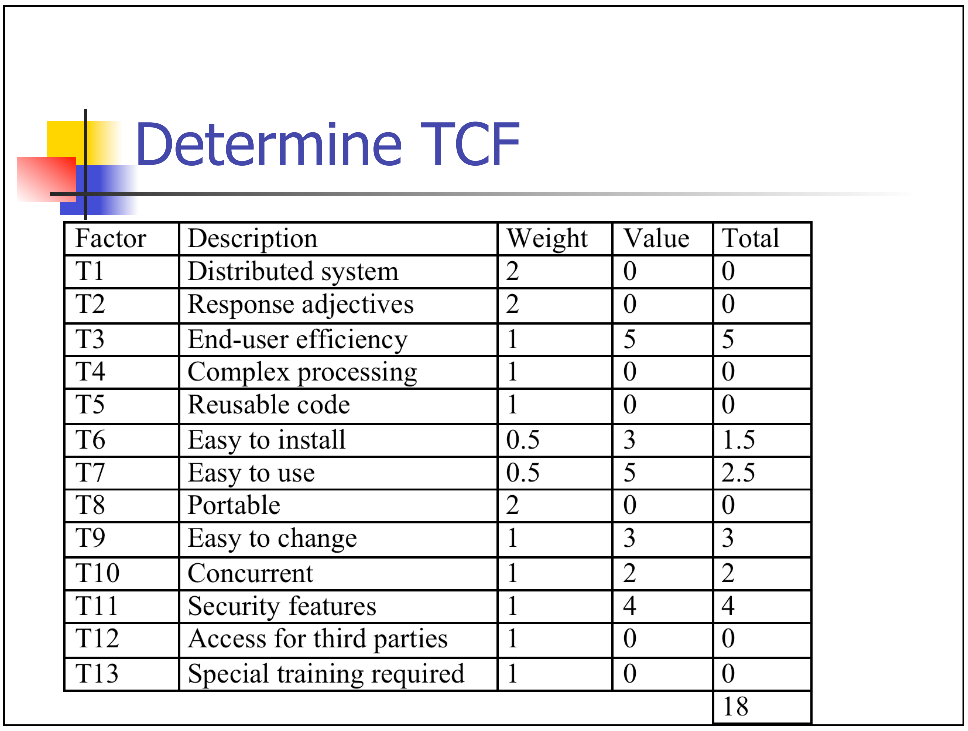
Video rental





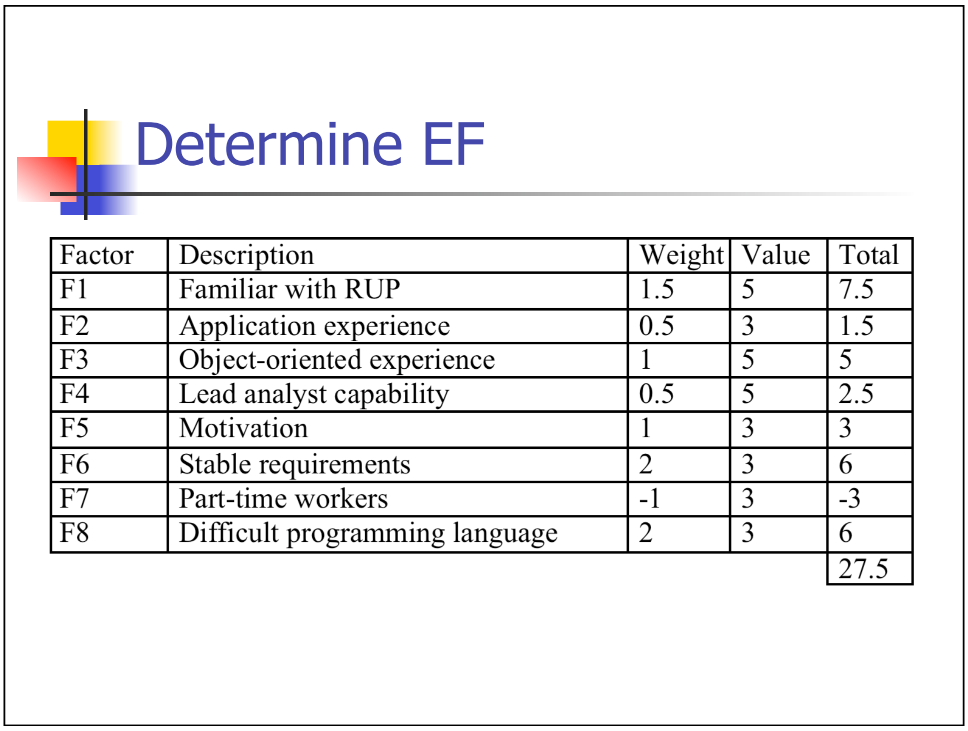
Determine UUCP

Use Case	Complexity (UUCP)
Membership	5
Find movie	5
Rent a movie	5
Movie reservation	5
Return a movie	5
Check late returns	5
Movie orders	5
Rental reports	5
Actors	
Member	3
Supplier	3
Employee	3
Store manager	3
Store owner	3
	55



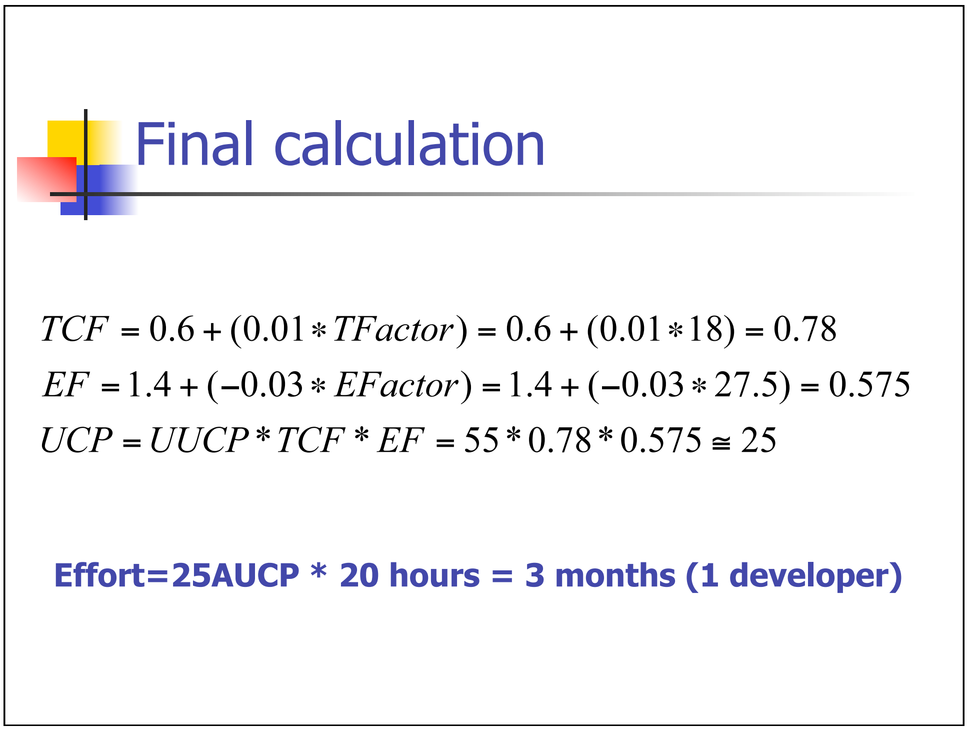
Determine TCF

Factor	Description	Weight	Value	Total
T1	Distributed system	2	0	0
T2	Response adjectives	2	0	0
T3	End-user efficiency	1	5	5
T4	Complex processing	1	0	0
T5	Reusable code	1	0	0
T6	Easy to install	0.5	3	1.5
T7	Easy to use	0.5	5	2.5
T8	Portable	2	0	0
T9	Easy to change	1	3	3
T10	Concurrent	1	2	2
T11	Security features	1	4	4
T12	Access for third parties	1	0	0
T13	Special training required	1	0	0
				18



Determine EF

Factor	Description	Weight	Value	Total
F1	Familiar with RUP	1.5	5	7.5
F2	Application experience	0.5	3	1.5
F3	Object-oriented experience	1	5	5
F4	Lead analyst capability	0.5	5	2.5
F5	Motivation	1	3	3
F6	Stable requirements	2	3	6
F7	Part-time workers	-1	3	-3
F8	Difficult programming language	2	3	6
				27.5



Final calculation

$$TCF = 0.6 + (0.01 * TFactor) = 0.6 + (0.01 * 18) = 0.78$$

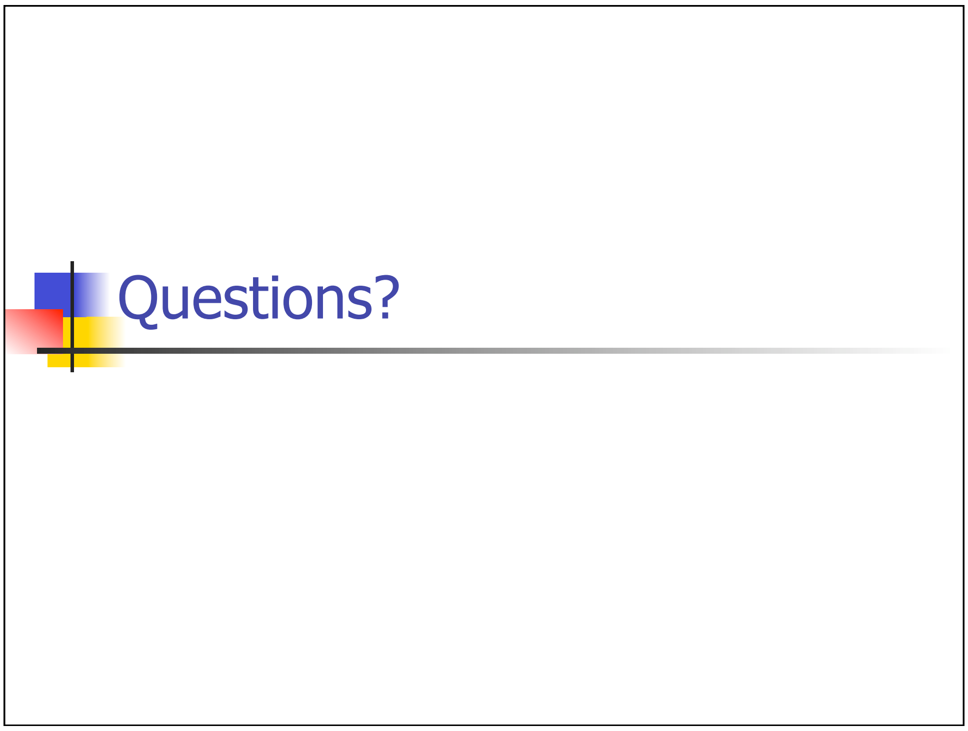
$$EF = 1.4 + (-0.03 * EFactor) = 1.4 + (-0.03 * 27.5) = 0.575$$

$$UCP = UUCP * TCF * EF = 55 * 0.78 * 0.575 \cong 25$$

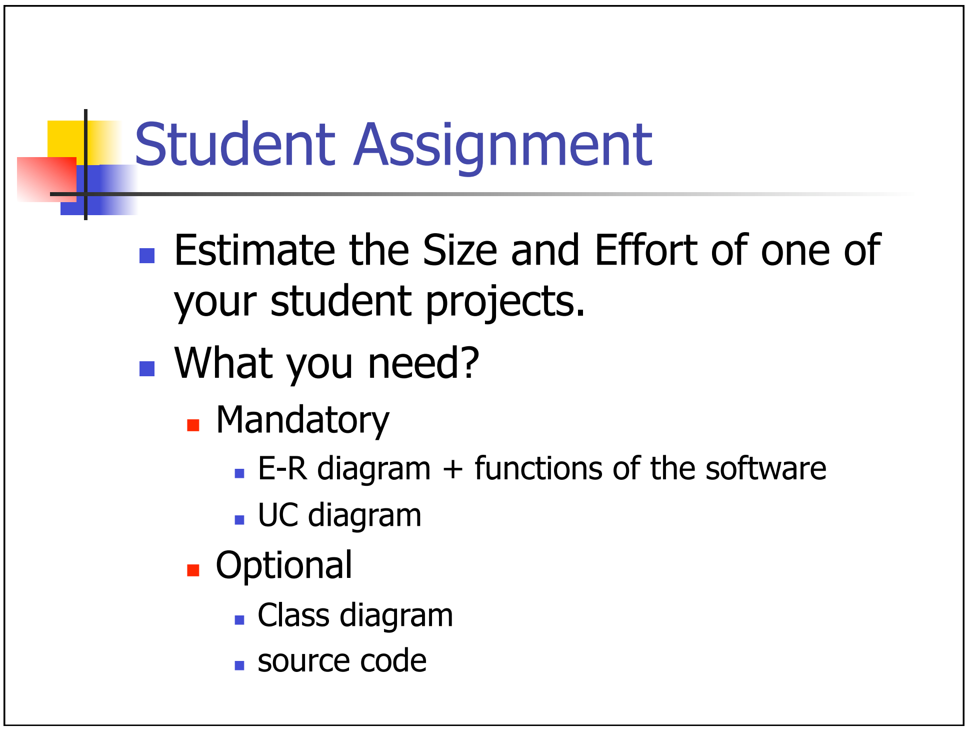
Effort=25AUCP * 20 hours = 3 months (1 developer)

Short demo

ISBSG REPOSITORY

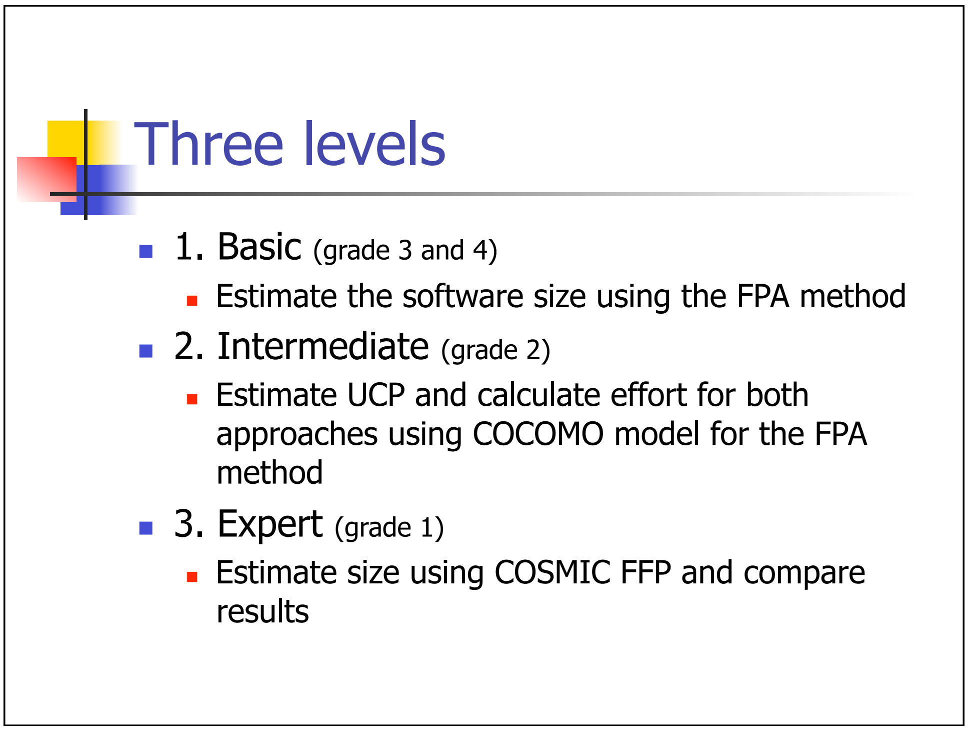


Questions?



Student Assignment

- Estimate the Size and Effort of one of your student projects.
- What you need?
  - Mandatory
    - E-R diagram + functions of the software
    - UC diagram
  - Optional
    - Class diagram
    - source code



Three levels

- 1. Basic (grade 3 and 4)
  - Estimate the software size using the FPA method
- 2. Intermediate (grade 2)
  - Estimate UCP and calculate effort for both approaches using COCOMO model for the FPA method
- 3. Expert (grade 1)
  - Estimate size using COSMIC FFP and compare results

Deadlines

- First deadline: 22nd of June
- Second deadline: 20 of July

- Send your assignments to:
  - ales.zivkovic@uni-mb.si

Problems

- If you don't have any projects:
  - ask your colleagues that did not take the course if they have one you can work on
  - use open source software
    - install the software and get functions from the user interface (you can also use User Manual if available)
    - reverse engineer database for the E-R diagram
    - use source code for LOC count