Master Degree Programme in Computer Science

Enterprise Information Systems

8. Software Size Estimation

SAPIENZA
UNIVERSITÀ DI ROMA

Prof. Ing. Claudio CILLI
cilli@di.uniroma1.it
http://dsi.uniroma1.it/~cilli

Introduction – Why?

- Industry has a reputation for software being over budget, late in delivery and low quality
- Unreasonable time and budget restraints force shoddy programming to maintain timeline
- Client satisfaction cannot be achieved with poor quality software even if it is on time and on budget
- To avoid this, means must be developed to accurately predict:
  - ***The product deliverables needed to satisfy the requirements***
  - ***The resources needed to generate the deliverables***
- Accurately predicting the size of a project results in better project planning

SAPIENZA UNIVERSITÀ DI ROMA

…Introduction – Why?

- Realistic allocations of time and resources can become possible with a good size estimate
  - ***"Development Effort is proportional to Size"***
- To analyze past projects and apply lessons learned from bad estimates
  - ***"You cannot manage what you cannot measure"***
- Provide project tracking to judge whether the project is progressing according to plan or whether changes must be made to the estimate

SAPIENZA UNIVERSITÀ DI ROMA

…Introduction – When?

- As early as possible in the project
  - ***"At no other time are the estimates so important than at the beginning of a project"***
- Estimation is an iterative process, occurring many times thoughout development
- After requirements, analysis, design, coding, testing, etc….
- Whenever new information that affects the project becomes available
  - ***"Predicting the size of a software system becomes progressively easier as the project advances"***

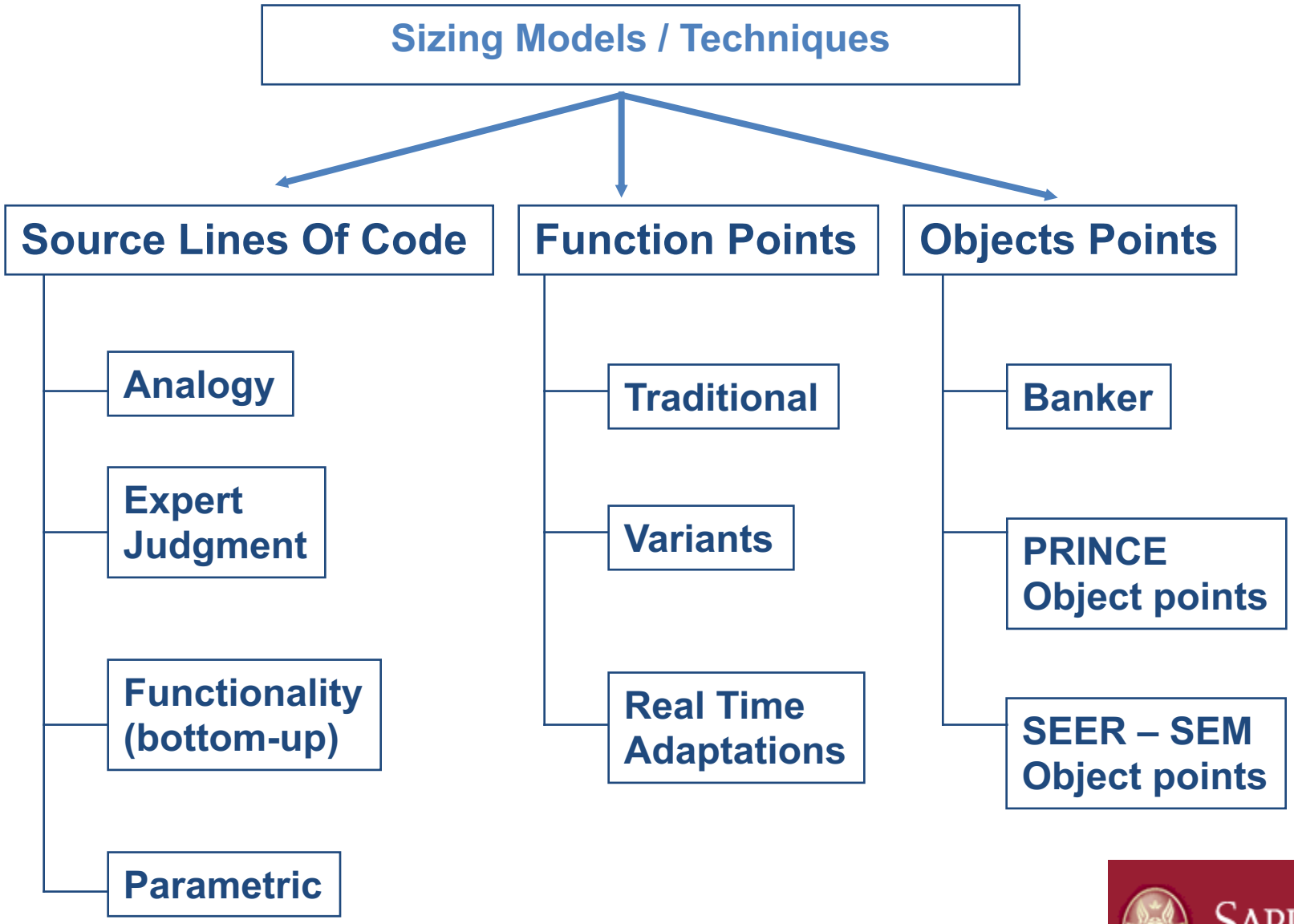
SAPIENZA UNIVERSITÀ DI ROMA

…Introduction – What?

- Different phases in development can be estimated:
  - ***Requirements Documents***
  - ***Design Documents***
  - ***Code***
  - ***Test Cases***
- Generally what is needed is an idea of the scope of the project
- Getting a feel for the size of project can indicate how much of everything else is needed

SAPIENZA UNIVERSITÀ DI ROMA

Size Estimation



SAPIENZA UNIVERSITÀ DI ROMA

SLOC – Analogy Model

- Software size, usually expressed in source lines of code (SLOC) is the key input to most software cost models
- The most popular measure of size is still SLOC, despite some of the advantages of function points and object points
- These models estimate size by comparing a program with a similar program or programs of know size

S = F x (size of similar packages)

  - *"S" is size (usually in SLOC)*
  - *"F" is a factor determined by experience or politics*

- i.e. Software Size Estimator (Singhal, 1986)

SAPIENZA UNIVERSITÀ DI ROMA

SLOC – Bottom-Up Model

- a.k.a Functionality Model
- These models compute SLOC from components of known size from a database
- In contrast to traditional bottom-up models, they are often useful early in a program since many functions are defined during software requirements analysis or preliminary design
- i.e. SASET cost model (Ratliff, 1993)
- It allows size to be input based on functionality. The user specifies the functions performed by the program, and the model obtains a SLOC estimate for each function from the database

SAPIENZA UNIVERSITÀ DI ROMA

SLOC – Expert Judgment Model

- These models estimate SLOC based on the opinions of one or more experts

$$S = (a+4m+b) / 6$$

  - *"S" is SLOC,*
  - *"a' is the smallest possible size*
  - *"m" is the most likely size*
  - *"b" is the largest possible size.*
  - *"a","m","b" is determined by experts*

- i.e. SEER Software Sizing Model (SEER-SSM, 1995), Developed by George Bozoki

- Steps
  1. First performs a pairwise comparison of program sizes.
  2. Inputs a Program Evaluation and Review Technique (PERT) estimate for all new programs.
  3. Selects the most probable size range from a set of ranges generated by the model.
  4. Performs a ranking to verify the pairwise comparison made initially

SAPIENZA UNIVERSITÀ DI ROMA

SLOC – Parametric Model

- These models use inputs consisting of numerical or descriptive values to compute program size in SLOC
- They are developed using regression analysis and other methods, such as numerical analysis or management surveys

Ikatura and Takayanagi Regression-Based Size Model

(Based on 38 COBOL Programs)

Candidate Parameters (For Y = SLOC, Comment Lines Not Counted)

X1 = Number of Input Files
X2 = Number of Input Items
X3 = Number of Output Files
X4 = Number of Output Items
X5 = Number of Items of Transaction Type Reports
X6 = Number of Vertical Items in Two-Dimensional Table Type Reports
X7 = Number of Horizontal Items in Two-Dimensional Table Type Reports
X8 = Number of Calculating Processes
X9 = Existence of Sorting
X10 = Sum of Output Items (X5+X6+X7)
X11 = Sum of Output Items In Files and Reports (X4+X5+X6+X7)

Best Fit Equation: $Y = -810 + 310X1 + 1.12X2 + 553X3 + 5.91X5 + 1.62X7 + 99.7X8$

SAPIENZA UNIVERSITÀ DI ROMA

SLOC - Analysis

- **Advantages**
  - Simple
  - Widely used
  - Most common size metric
  - Directly relates to the end-product
  - Counting can be automated
- **Disadvantages**
  - No established standard of LOC
  - Language-dependent
  - Hard to visualize early
  - Hard to understood by clients
  - Does not address complexity or other environmental factors
  - Reactive measure

SAPIENZA UNIVERSITÀ DI ROMA

Function Points (FP)

- Function point models were popularized by the extensive research of Albrecht and Gaffney (Albrecht and Gaffney, 1983)
- They estimated the size of software in terms of functionalities from users' viewpoint
- Function points are not only a valid predictor of software size, but also were superior to SLOC as a predictor of development cost effort

SAPIENZA UNIVERSITÀ DI ROMA

FP – Traditional Models

- "Basic" Function Points:

BFP = 4 EI + 5 EO + 4 EQ + 10 ILF + 7 EIF

  - *External inputs (EI): input screens and tables*
  - *External outputs (EO): output screens and reports*
  - *External inquiries (EQ): prompts and interrupts*
  - *Internal files (ILF): databases and directories*
  - *External interfaces (EIF): shared mathematical routines*
  - *(With + / - 25% Complexity Adjustment)*

SAPIENZA UNIVERSITÀ DI ROMA

FP – Traditional Models

- **Unadjusted Function Points (UFP)**: Weight Five Attributes

Attributes	**Complexity Weighting**			**Total**
	Simple	Average	Complex	
EI	3	4	6	
EO	4	5	7	
EQ	7	10	15	
ILF	5	7	10	
EIF	3	4	6	

SAPIENZA UNIVERSITÀ DI ROMA

FP – Traditional Model

- **Adjusted Function Points (FP)** = UFP x (0.65 + [0.01 x CA])
- *(CA is Complexity Adjustment; Sum of 14 Factors, Rated 1 to 5 for Influence [0 - None, 1 - Little, 2 - Moderate, 3 - Average, 4 - Significant, 5 - Strong];*

Data Communications	On-Line Update
Distributed Functions	Complex Processing
Performance	Re-Usability
Heavily Used configuration	Installation Ease
Transaction Rate	Operational Ease
On-Line Data Entry	Multiple Sites
End User Efficiency	Facilitate Change

SAPIENZA UNIVERSITÀ DI ROMA

FP - Variants

- There have been several attempts to modify the traditional or Albrecht function points
- CHECKPOINT model, QSM Size Planner (QSM, 1987) do not use complexity adjustment factors; however, they use five attribute ratings instead of three
- Symon's Mark II Function Points (Symons, 1991). Mark II function points use only three attributes; inputs, outputs, and entities

SAPIENZA UNIVERSITÀ DI ROMA

FP – Variants: Real Time Adaptation

- Although function points were originally researched for business applications, there have been several attempts to adapt function points to scientific and real-time programs

- **Feature Points**: Feature points add a sixth attribute, algorithms (AL), to the five used in traditional function points. According to Capers Jones, an algorithm is a "set of rules which must be completely expressed in order to solve a computational problem" (Jones,1991)

  The basic feature points (BFP) equation is: BFP = 3AL + 4EI + 5EO + 4EQ + 7ILF + 7 EIF

- **ASSET-R Function Points**: The ASSET-R model (Reifer, 1989) uses three additional attributes to those used in traditional function points: **operating modes, rendezvous, and stimulus/response relationships:**
  - Operating modes are time-dependent end-to-end processing flows; rendezvous are a measure of concurrency in real-time systems; and stimulus/response relationships measure the amount of sequencing and control in real-time systems

SAPIENZA UNIVERSITÀ DI ROMA

FP-to-SLOC Conversion

- It is sometimes necessary to convert from SLOC to function points, or vice-versa
- Such as COCOMO (Constructive Cost Model), CHECKPOINT (Software Cost Estimation)

Language	Jones (1996) SLOC/FP	Reifer(1986) SLOC/FP
Assembler	320	400
COBOL	107	100
FORTRAN	107	105
Ada	71	72
PROLOG	64	64
Pascal	91	70

SAPIENZA UNIVERSITÀ DI ROMA

FP - Analysis

- **Advantages**
  - Easier to make early estimates
  - Independent from implementation language, developer experience
  - Established standard (IFPUG)
- **Disadvantages**
  - Hard to count automatically
    - Time consuming
  - Complexity factors can be judgmental
    - Repeatability
    - Depends on estimator's experience

SAPIENZA UNIVERSITÀ DI ROMA

Object Points

- Develop by Banker et al. [Banker et al]
- Originally intended for CASE tool development projects
  - May apply to other projects
- Adapted for Object-Oriented software
- Use object counts instead of function counts
- An “object” can be:
  - Logical system component (**Rule Sets**)
  - Language-specific constructs (**3GL-Module**)
  - UI component (**Screen Definitions**)
  - User report (**Report**)
- An “object” is NOT just a raw object class
- Conducted at a more macro level than Function Points

SAPIENZA UNIVERSITÀ DI ROMA

Object Points - How

Object Type	Object Complexity		
	Simple	Medium	Difficult
Rule Sets			
3GL-Module			
Screen Definition			
User reports			

- Steps
  1. For each object type, count all instances
  2. Each object is assessed a complexity weight

SAPIENZA UNIVERSITÀ DI ROMA

Object Points - How

- Steps
  3. Sum up complexity weights of all objects to get the Object-Point (OP)
  4. Multiply OP by a reuse factor (RF)
  5. If this is not a new program, calculate NOP (New Object-Point)

NOP = OP (1 – RF)

SAPIENZA UNIVERSITÀ DI ROMA

Object Points - How

- Steps
  6. Based on NOP, determine expected productivity cost

Developers Experience, ICASE maturity / capability	Very Low	Low	Nominal	High	Very High
Productivity cost	4	7	13	25	50

Effort = NOP / Productivity cost

SAPIENZA UNIVERSITÀ DI ROMA

Object Points - Variants

- Other variations exist
  - PRICE Object-Points [Ferens]
  - SEER-SEM Object-Points [Ferens]
  - WebObjects [Reifer]
- No established standards
- Some variants have rules for OP-to-FP conversion

SAPIENZA UNIVERSITÀ DI ROMA

Object Points - Analysis

- **Advantages**
  - Same as in Function Points
  - May be easier to visualize and understood by clients
- **Disadvantages**
  - Less use than Function Points
  - Lack of standards

SAPIENZA UNIVERSITÀ DI ROMA

Wideband-Delphi

- Can be applied at the project or component level
- Usually only examine a small section or component of the overall effort/project [Boehm]

- Steps:
  1. A group of experts is each given the program's specification and an estimation form
  2. The group of experts and a moderator meet to discuss the product and any estimation issues
  3. Each estimator anonymously complete the estimation forms
  4. Moderator collects completed estimation forms
     - tabulates the results
     - returns them to the experts
     - Only each estimator's personal estimate is identified; all others are anonymous
  5. Experts meet to discuss the results, revising their estimates as they feel appropriate
  6. Repeat Step #1-5 until the estimates converge to an acceptable range

SAPIENZA UNIVERSITÀ DI ROMA

Wideband-Delphi

- **Advantages**
  - Iterative, team based, collaborative estimating
  - Less biased than individual estimation
  - Does not require historical data
  - Can be used at both high-level and detailed level estimation

- **Disadvantages**
  - May be hard to find more than one expert
  - Difficult to repeat with different group of experts
  - Possible to reach consensus on an incorrect estimate, people may not be skeptical enough
  - Can develop a false sense of confidence
  - May fail to reach a consensus
  - Experts may be biased in the same subjective direction
  - Lots of overhead (time, team involvement, planning) for a relatively small sets of tasks
  - Takes quite a few “steps” or iterations
  - Does not require historical data

SAPIENZA UNIVERSITÀ DI ROMA

PROBE

- **PRO**xy **B**ased **E**stimation

- Proxy are used instead of SLOC
  - Function Points
  - Object-Points
  - Other levels of granularity (slides in a presentation, pages in a book, etc.)

- Requires historical data

- A good proxy should be:
  - Related closely to the effort required to develop the product
  - Automatically counted
  - Visualized easily and early in the project
  - Customizable according to organization needs

SAPIENZA UNIVERSITÀ DI ROMA

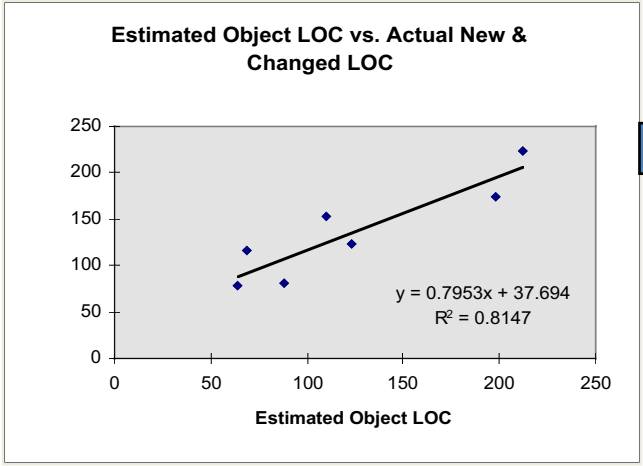
PROBE

Choose a Proxy → **Conceptual Design (Count Proxy)**

Historical Proxy-to-LOC data

Conceptual Design (Count Proxy) → **Calculate Est. Proxy-LOC**

Calculate Est. Proxy-LOC → **Linear Regression Analysis**



Linear Regression Analysis → **Regression Parameters**

Regression Parameters

β_0 = ? β_1 = ? r^2 = ?

Est N&C LOC = β_0 + β_1 * Est Proxy-LOC

SAPIENZA UNIVERSITÀ DI ROMA

Size Estimation Methods - PROBE

- **Advantages**
  - Forces you to keep and use historical data
- **Disadvantages**
  - Doesn't work without historical data
  - Not always have appropriate historical data
  - Not always have enough appropriate historical data
  - Needs lots of historical data for linear regression analysis to work

SAPIENZA UNIVERSITÀ DI ROMA

Reliable Size Estimations

- **Attributes:**
  - Structured
  - Defined
  - Applicable for all projects
  - Applicable throughout a project
  - Easily adjustable for future projects
  - Susceptible to statistical analysis
  - Use of a suitable estimation proxy
  - Ability to automate

SAPIENZA UNIVERSITÀ DI ROMA

What size estimates are NOT

- Often programmers and managers misinterpret what size estimates are for. They are not:
  - Upper limits that programmers should be working towards or down to
  - A minimum estimate set by project managers to win a contract

SAPIENZA UNIVERSITÀ DI ROMA

Problems with Size Estimation

- Size estimation is relatively new, not a science
- Lack of historical data leads to:
  - Gut instincts
  - Flawed Project Plans
  - Unhappy customer and programmers
- Estimates are only as good as the inputted data
- Estimates tend to be overly optimistic
- Very complex, error prone and highly experience dependent (whether historical or expert)
- Most software is unique: environment, industry, tools, architectures, user loads, performance and usability expectations

SAPIENZA UNIVERSITÀ DI ROMA

References

[Banker et al] R. Banker, R. Kauffman, C. Wright, D. Zweig. (1994). Automatic Output Size and Reuse Metrics in a Repository-Based Computer-Aided Software Engineering (CASE) Environment. IEEE Transactions on Software Engineering. 20(3)

[Boehm et al] B. Boehm, C. Abts, B. Clark, S. Devnani-Chulani. (1997). COCOMO II Model Definition Manual. The University of Southern California

[Boehm] B. Boehm. (1981). Software Engineering Economics. Prentice Hall.

[Fenton et al] N. Fenton, S. Pfleeger. (1997). Software Metrics: A Rigorous and Practical Approach. International Thomson Computer Press

[Ferens] D. Ferens. (1999). Software Size Estimation: Quo Vadis?. National Estimator

[Reifer] D. Reifer. (2000). Web Development: Estimating Quick-to-Market Software IEEE Software. 17(6)

SAPIENZA UNIVERSITÀ DI ROMA