COSC 310: Software Engineering

Dr. Bowen Hui
University of British Columbia Okanagan

Recall Estimation from PM Lecture

- prediction of resources needed to complete a project based on predicted size
  - estimate **work effort**
- cost is always estimated in terms of effort, not actual dollars
- considers:
  - calendar time
  - staffing availability, capability, productivity
  - budget constraints
  - creeping requirements
  - what-if analysis

Fundamental Estimation Questions

- how much work is required to be produced in order to complete an activity?
- how much effort and calendar time are needed to complete an activity?
- what is the total cost of an activity?
  - this is monetary cost: hardware, software, travel, training, human resources, insurance

"Best" Estimates

- there is no simple way to make accurate estimate of effort required to develop software
- initial estimates often based on:
  - inadequate requirements
  - unfamiliar technology
  - limited information about development team
- project cost estimates may be self-fulfilling
  - estimates defined to meet expected budget

Brief History

First Formal Methods 1970's

- PRICE S: First computerized estimation model
- SLIM: Precursor to COCOMO; based on 50 US military projects
- Function Point Analysis: Developed by IBM

COCOMO 1981

- Developed by Barry Bohem of TRW Aerospace
- Designed to estimate development of large, batch-oriented systems

COCOMO II 1997

- Reengineered for development of desktop systems
- Requires estimation of entire project size & 21 "adjustment factors"
- Current estimation tools based on COCOMO II; static for 10+ years

Agile 2001

- Time-based development estimates
- Iteration focused; less attention to overall project estimation

Size-Based vs Time-Based Estimates

Size-based estimates

- How many lines of code will you write?
- How many classes, tables, screens, interfaces?
- How many function points will you implement?

Challenges:

- Very hard to do – most developers don't think this way
- Considers only development aspect of project

Time-based estimates

- How much effort will it take to complete story or task?

Challenges:

- Developers underestimate – or do they?

COCOMO

- stands for constructive cost model
- data collected from 63 software projects
  - actual LOC
  - actual effort
  - actual schedule duration
- **regression analysis** was used to develop equations that best describe the data
  - a statistical technique used to find relationships between variables for predicting future values
- several COCOMO models have been developed
  - Ex: agile COCOMO
  - http://sunset.usc.edu/research/cocomosuite

(Basic) COCOMO Overview

- 3 modes (project types)
  - organic, semi-detached, embedded
- 3 levels of analysis (varies in accuracy)
  - basic, intermediate, detailed
- idea:
  - estimate effort
  - estimate development time
  - estimate staffing needs
  - estimate productivity
  - extension to allow for **cost drivers** and complex system structure

Modes (Project Types)

1. Organic Mode
   - characterized by small team size, familiar environment little innovation, few constraints and deadlines and well-understood applications such as payroll, inventory and so on
2. Semi-Detached Mode
   - characterized by team members that are mix of experienced and inexperienced, partly familiar with the system being developed such as compliers
3. Embedded Mode
   - characterized by complex (real-time) systems, diverse nature of projects such as air-traffic control, weapon systems

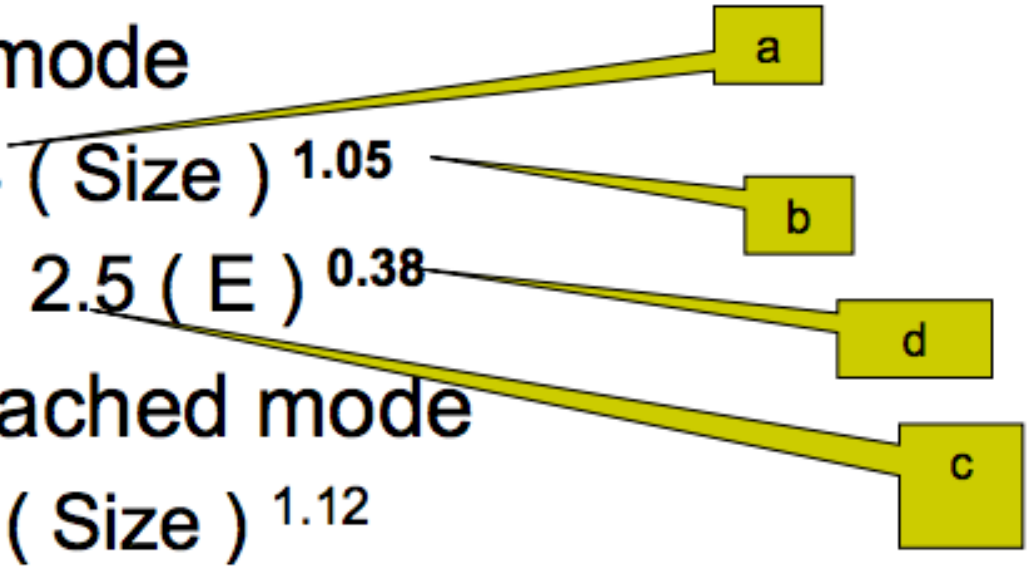
Basic Analysis

- Estimate effort
  - $E = a \times Size^b$ Size in KLOC; E in staff-months
- Estimate dev time
  - $TDEV = c \times E^d$ TDEV in months
- Estimate staffing
  - SS = E / TDEV SS in average num. employees
- Estimate productivity
  - P = Size / E P in (K)LOC per staff-months

- Use different empirical constants for each mode
- Model provides generic constants for organizations without much data (next slide)

Prescribed Constants

- Organic mode
  - $E = 2.4 (\text{Size})^{1.05}$
  - $TDEV = 2.5 (E)^{0.38}$
- Semi-detached mode
  - $E = 3.0 (\text{Size})^{1.12}$
  - $TDEV = 2.5 (E)^{0.35}$
- Embedded mode
  - $E = 3.6 (\text{Size})^{1.2}$
  - $TDEV = 2.5 (E)^{0.32}$



Example of Basic Analysis

- Delivered Source Instructions = 32000 LOC (32KLOC)
- Project Category = Organic Mode
- $E = 2.4 (\text{Size})^{1.05}$
  - $= 2.4 (32)^{1.05}$
  - = 91 staff-months
- $TDEV = 2.5 (E)^{0.38}$
  - $= 2.5 (91)^{0.38}$
  - = 14 Months

Example (cont.)

- Staff Size = Effort / TDEV
- = 91/14
- = 6.5
- Productivity = Size / Effort
- = 32000/91
- = 352
- A well-understood project with a small team (6.5 SS) and known technology of product size (32KLOC) requires 14 months to build (effort = 91 staff-months), assuming a productivity rate of 352 LOC per staff-month and a team size of about 6-7 members.

Intermediate Analysis

- Accounts for environmental factors in which the software is developed
- So the effort needs to be adjusted
- The factors that either increase or decrease the effort are called Cost Drivers
- These are classified into:
  - Product Attributes
  - Computer Attributes
  - Personnel Attributes
  - Project Attributes
  - Other drivers

Effort Adjustment Factor (EAF)

- $E = a (Size)^{b}$ - Basic analysis
- Adjusted E = EAF * E – Intermediate analysis
  - where EAF is the effort adjustment factor based on set of cost drivers
- $EAF = C_1 \times C_2 \times C_3 \times \ldots C_n$ where n= 15

- $C_i = 1$ cost driver does not apply
- $C_i > 1$ implies increased cost due to this factor
- $C_i < 1$ implies decreased cost due to this factor

Cost Drivers - Product Attributes

- Some of the product attributes considered are :
  - Reliability (real-time applications) -- RELY
  - Database Size (data processing applications) --DATA
  - Product Complexity (execution time constraints) --CPLX
- These attributes are rated on a scale ranging from very low.... extra high
  - very low, low, nominal, high, very high, extra high
  - E.g., RELY = very low, C_1 = .88
  - E.g., CPLX = extra high, C_3 = 1.65

Remaining Cost Drivers

- Computer Attributes (platform) include
  - execution time constr. (processor speed) -- TIME
  - storage constraints (main memory size) -- STOR
  - turn around time (not applicable in current hw/sw)--TURN
  - Virtual machine volatility (HW/OS on target machine) ---VIRT
- Personnel Attributes include:
  - Analyst capability --ACAP
  - Application Experience --AEXP
  - Programmer Capability -- PCAP
  - Programming Lang. Experience --LEXP
  - Virtual Machine Capability (includes OS and HW) --VEXP
- Project Attributes
  - Use of CASE Tools -- TOOL
  - Modern Programming Practice (OO/structured tech.)--MODP
  - Project Development Schedule (accelerated?) -- SCED

Intermediate Analysis Effort Calculation Steps

- Calculate E using Basic Analysis
- Determine cost driver values and calculate EAF
- Calculate adjusted E using new constants:
  - $E = EAF \times 3.2 (Size)^{1.05}$ Organic Mode
  - $E = EAF \times 3.0 (Size)^{1.12}$ Semi-detached Mode
  - $E = EAF \times 2.8 (Size)^{1.2}$ Embedded Mode
- TDEV, SS, P equations remain the same

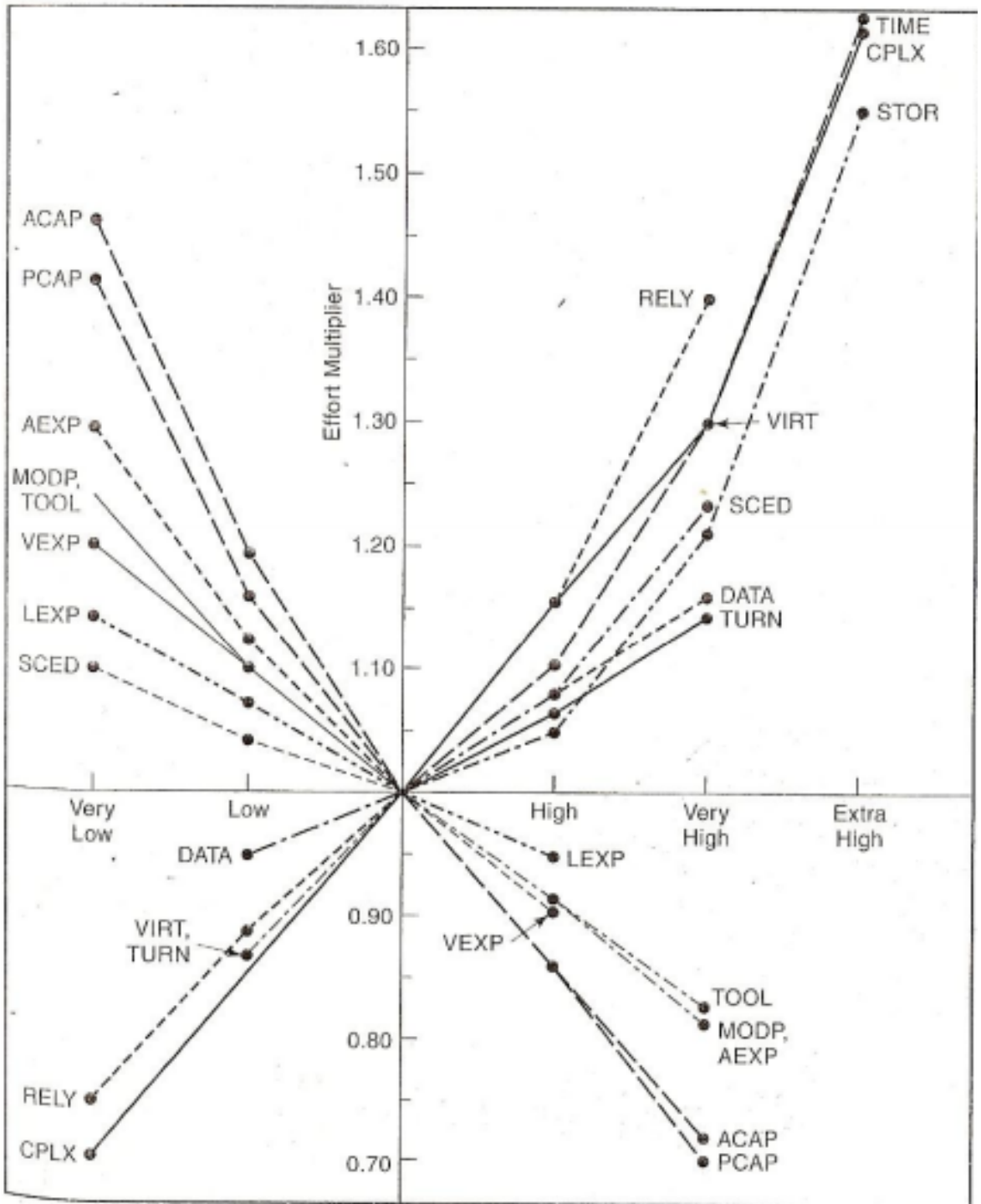
- NOTE : Intermediate Model replaces the basic model. Parameters a,b have been recalibrated

EAF Example

- Delivered Source Instructions = 128000
- Project Category = Embedded Mode
- Effort = 2.8 (128) $^{1.20}$
  - = 875 staff months
- Reliability= Very low (Multiplier = .75)
- Complexity= Very low (Multiplier = .7)
- Memory Limitation =none (Multiplier = 1)
- Tool Use = high (Multiplier = .9)
- Schedule= normal (Multiplier = 1)
- Effort = 875*.75*.7*1* .9* 1
  - = 413 staff months

Comments on Cost Drivers

- Based on cost drivers, the estimates varied from 2312 (high) to 413 (low) in our example
- Who decides on this multiplier values?
  - The PM team has to classify the attribute as well as decide on the exact value of the multiplier
  - It is a subjective estimate
- Other cost drivers can be added: security requirements, access to data, requirements volatility..
- Limitations:
  - Software Product is considered as a single entity and multipliers are applied to the entire product


Effort Multiplier
1.60
1.50
1.40
1.30
1.20
1.10
0.90
0.80
0.70
Very Low
Low
High
Very High
Extra High
TIME
CPLX
STOR
RELY
VIRT
SCED
DATA
TURN
LEXP
TOOL
MODP, AEXP
ACAP
PCAP
VEXP
ACAP
PCAP
AEXP
MODP, TOOL
VEXP
LEXP
SCED
DATA
VIRT, TURN
RELY
CPLX

Detailed Analysis

- System, subsystem, modules
- Cost drivers are analyzed separately
- Subsystem inherit system cost drivers
- Modules inherit the subsystem cost drivers

- Ref: Software Engineering Economics by Barry Boehm, Prentice-Hall, 1981.
- Software Cost Estimation with COCOMO II by Boehm et al, Prentice-Hall, 2000.

COCOMO Pros and Cons

- Pros:
  - ?

COCOMO Pros and Cons

- Pros:
  - a repeatable process
  - back fitted from real data
  - allows new cost drivers
  - supports different modes and analysis levels
- Cons:
  - ?

COCOMO Pros and Cons

- Pros:
  - a repeatable process
  - back fitted from real data
  - allows new cost drivers
  - supports different modes and analysis levels
- Cons:
  - assumes waterfall SDLC
  - assumes simplistic view of lifecycle phases
  - original model divided effort as 30% design, 30% code and unit test, 40% integration
  - all analysis levels are size dependent

COCOMO II

- accounts for different SDLCs
- allows for FP estimates (rather only LOC)
- supports different models:
  - application composition
    - suitable for GUI builder tools
    - based on **object points** (not FPs)
  - early design
    - obtain rough estimates of costs and duration without system architecture
    - based on unadjusted FP
  - post architecture
    - most detailed of all
    - use after architecture is developed

Estimation in Agile Projects

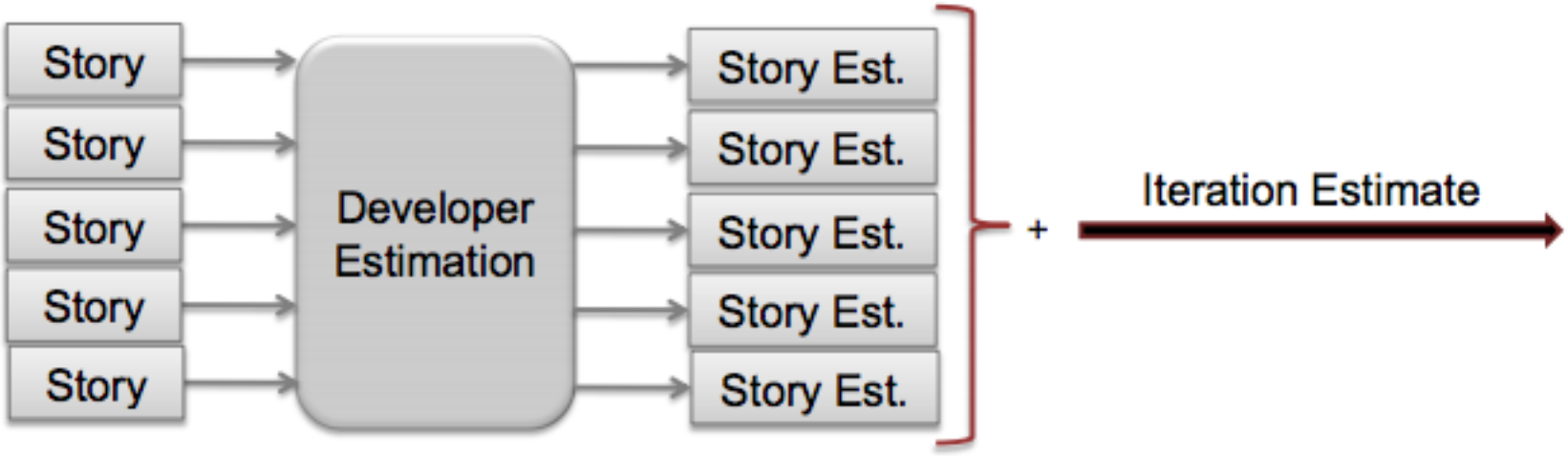
Since estimation is difficult

- Use time/effort-based estimates
- Do not estimate the future any further than is necessary
- Rely on yesterday's weather to forecast
- Shorten time between estimation and feedback

Shortcomings

- Simplistic methods = under-estimation
- Iteration-based, not project-based
- Budget, resources and schedule **are** estimated for most Agile projects – but not by team doing work

Current Agile Estimation Methods



Simple time estimate inputs:

- Not a dark art
- Reproducible

Output is inaccurate and incomplete:

- Not project-based
- Schedule, Resources, Costs??
- Cumulative error = Under-estimation

Summary Challenges with Current Methods

- Defined Estimation uses size-based inputs (LOC) and applies statisical formulae to derive estimates
  - Size-Based Estimation is hard to do and is error-prone
  - Numerous input paramteres add complexity
- Defined Estimation methods consider only development
- Agile leverages simple time-based estimation but sums these estimates incorrectly leading to under-estimation
- Agile focusses on the iteration, not the project

Midterm Format

- midterm next Friday
- 80 minutes
- cheatsheets allowed
- no calculators
- 8 questions
- each question: shows sub-questions and associated points

- review next class