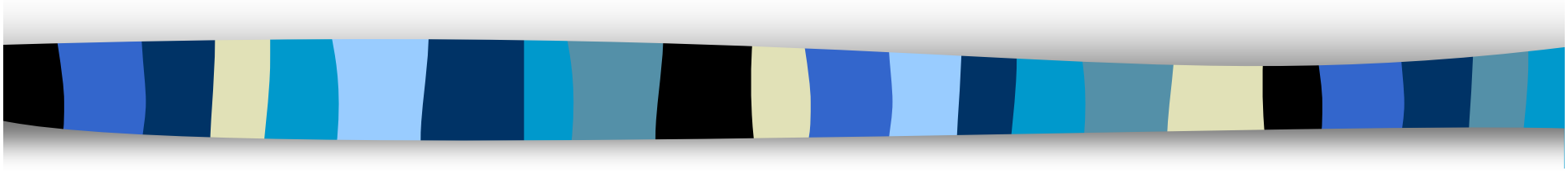
INF 111 / CSE 121: Software Tools and Methods



Lecture Notes for Fall Quarter, 2007

Michele Rousseau

Set 23

Some slides adapted from Susan E. Sim

Announcements

- **Quiz #4 and Assignment #3 grades.**
  - Be sure to pick up your Quiz #4 and Assignment #3 from the distribution center as soon as it is released – re-grade requests need to be in by next Friday.
- **Assignment #3 changes**
- **Readings:**
  - Van Vliet Chapter 7

Previously in INF 111…

- **You had a quiz… no review today**

Today's Lecture

- **Effort Estimation**

Better Estimation Techniques

- Estimating based on experience or hard data
  - **Expert judgment**
  - **Estimation by analogy**
- Variation: Delphi method
- Algorithmic cost modeling
- Personal Software Process

Expert Judgment

- One or more experts in both software development and the application domain use their experience to predict software costs.

- **Advantages**
  - Relatively cheap estimation method
  - Can be accurate if experts have direct experience with similar systems

- Disadvantages
  - Very inaccurate if there are no experts
    - **Are you an expert?**
  - Does not use hard data

Estimation by Analogy

- The cost of a project is estimated by comparing the project to a similar project in the same application domain

- Advantages
  - Accurate if project data available

- Disadvantages
  - Impossible if no comparable project has been undertaken
  - Estimates can be inaccurate if details overlooked
  - Subsequent similar projects can be quicker

Delphi Method

- Idea: Create a group expert opinion, while counterbalancing personality factors in process

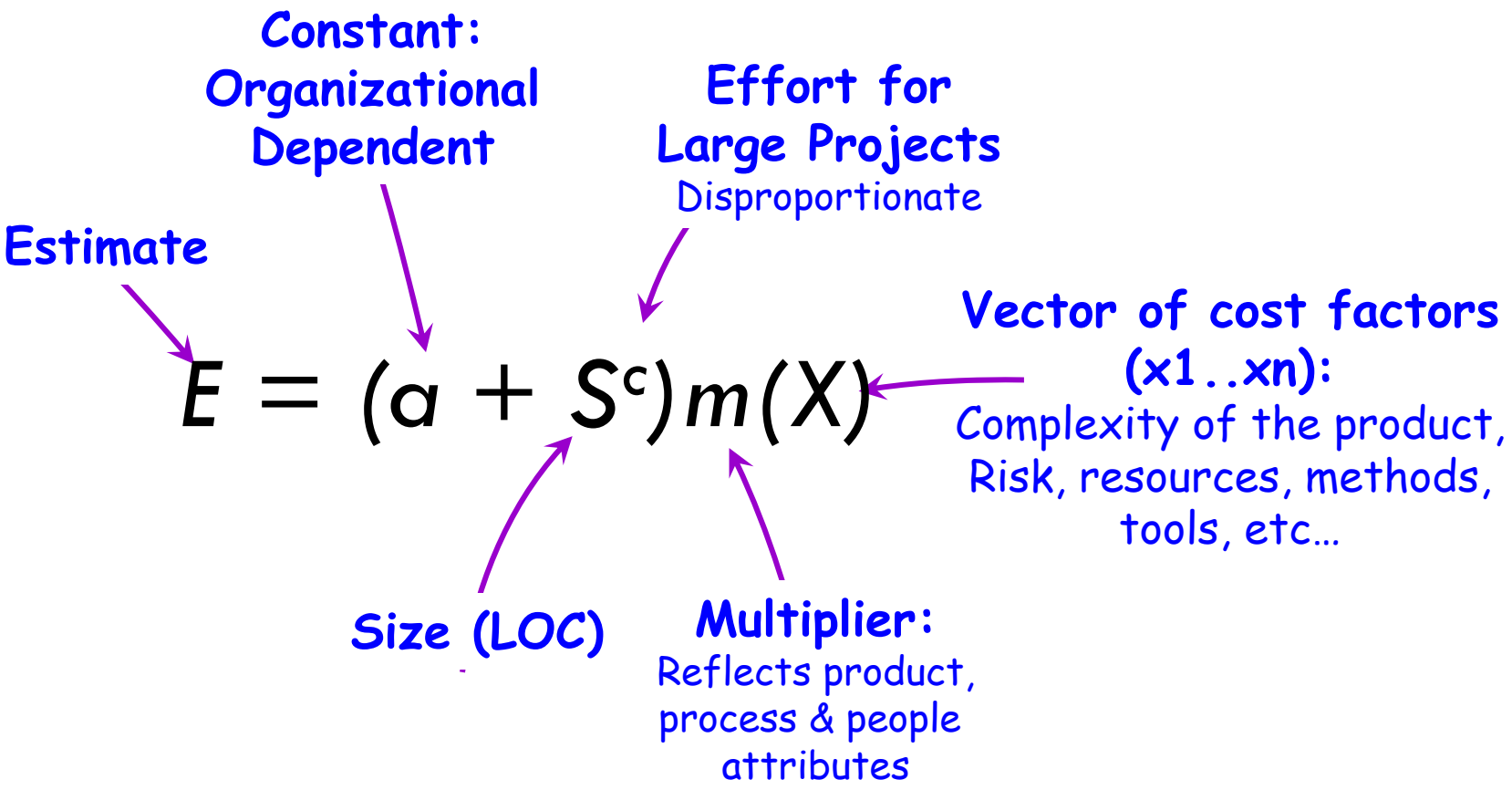
- Panel of independent expert estimators + moderator

1. Experts independently create estimates.
2. Moderator collects written estimates from individuals.
3. Estimates are distributed to group.
   - Anonymously
4. Experts deliver new estimates based on new information from moderator (others opinions may help fill in forgotten details)
5. Continue until consensus is reached.

Algorithmic Cost Modeling

- Cost and development time for a project is estimated from an equation
- Equations can come from research or industry
  - Analysis of historical data
  - Work best if they are tailored using personal and organizational data
    - **Adjust weights of metrics based on your environment**

Basic Equation

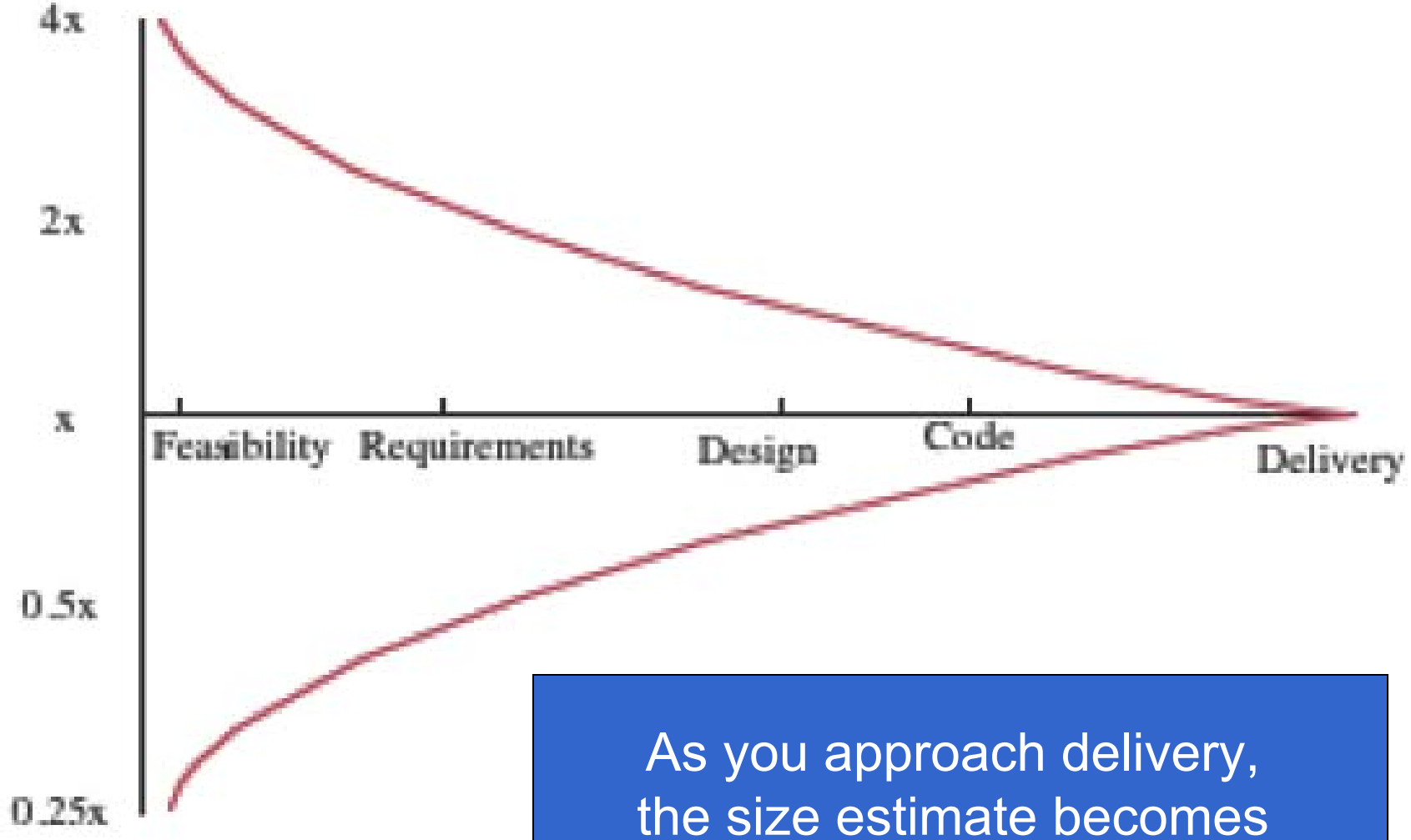


- Most commonly used product attribute for cost estimation is code size
- Most models are basically similar but with different values for a,c, & m

Problems with Algorithmic Estimation

- Effort estimates are based on size
  - **Highly inaccurate at start of project**
  - **Size is usually given in lines of code**
- Lines of code does not reflect the difficulty
  - **Some short programs are harder to write than long ones**
  - **Lines of code ≠ effort**
    - **Not all activities produce code**
  - **Programming Language: Java vs. assembler**
  - **Number of Components**
  - **Distribution of the system**
- Recall Brooks Chapter 2
  - **Effort ≠ Progress**
  - **The c exponent is an attempt to account for communication and complexity costs, but basic problem remains**

Estimate Uncertainty



As you approach delivery, the size estimate becomes more accurate

Example: COCOMO (Boehm)

Constructive Cost Model (COCOMO)

- COCOMO - one of the most widely used software estimation models in the world
- Empirical model based on project experience
- Well-documented, 'independent' model
  - not tied to a specific software vendor
- Long history
  - initial version published in 1981 (COCOMO-81)
- COCOMO II takes into account different approaches to software development, reuse, etc.
- Predicts the effort and schedule
  - based on inputs relating to the size of the software &
  - a number of cost drivers that affect productivity

COCOMO: Three Models

- **3 Models reflect the** complexity**:**
  - the Basic Model
  - the Intermediate Model
  - and the Detailed Model

The Development Modes: Project Characteristics

- Organic Mode
  - developed in a **familiar**, stable environment,
  - **similar** to the previously developed projects
  - relatively **small** and requires little innovation
  - Eg. Payroll system
- Semidetached Mode
  - **intermediate** between Organic and Embedded
  - Eg. Banking System
- Embedded Mode
  - tight, **inflexible** constraints and interface requirements
  - The product requires **great innovation**
  - Eg. Nuclear power plant system

Basic COCOMO Model:

Estimates the software development effort using only a *single predictor variable* (size in DSI) and 3 development modes

- When Should You Use It ?
  - Good for quick, early, rough order of magnitude estimates of software costs

Basic COCOMO Model: Equations

Mode	*Effort*	*Schedule*
Organic	$E=2.4*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semi-detached	$E=3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=3.6*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

Basic COCOMO Model: Example

- We have determined our project fits the characteristics of Semi-Detached mode
- We estimate our project will have 32,000 Delivered Source Instructions (DSI).

Using the formulas, we can estimate:

- Effort = $3.0*(32)^{1.12}$ = 146 man-months
- Schedule = $2.5*(146)^{0.35}$ = 14 months
- Productivity = 32,000 DSI / 146 MM
  = 219 DSI/MM
- Average Staffing = 146 MM /14 months
  = 10 FSP

Basic COCOMO Model: Limitations

- Its accuracy is necessarily limited because of its lack of factors which have a significant influence on software costs

- Estimates are within a factor of…
  - **1.3** only **29%** of the time **&**
  - **2** only **60%** of the time

Intermediate COCOMO Model

Estimates effort by using fifteen cost driver variables besides the size variable used in Basic COCOMO

- When should you use it?
  - Can be applied across the entire software product for easy and rough cost estimation during the early stage
  - or it can be applied at the software product component level for more accurate cost estimation in more detailed stages

Cost Drivers

Four areas for drivers

- **Product Attributes**
  - Reliability, Database Size, Complexity
- **Computer Attributes**
  - Execution Time Constraint, Main Storage Constraint, Virtual Machine Volatility, Computer Turnaround Time
- **Personnel Attributes**
  - Analyst Capability, Applications Experience, Programmer Capability, Virtual Machine Experience, Programming Language Experience
- **Project Attributes**
  - Modern Programming Practices, Use of Software Tools, Required Development Schedule

Subjective Assessments

Intermediate Model: Effort Multipliers

- Table of Effort Multipliers for each of the Cost Drivers is provided with ranges depending on the ratings

Cost Driver	*Very Low*	*Low*	*Nom*	*High*	*Very High*	*Extra High*
Product Complexity	**0.70**	**0.85**	**1.00**	**1.15**	**1.30**	**1.65**

Intermediate Model: Equations

Mode	*Effort*	*Schedule*
Organic	$E=EAF*3.2*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semi-detached	$E=EAF*3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=EAF*2.8*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

COCOMO Effort Equation

Effort = 3.0 * EAF * $(KSLOC)^E$

- Result is in Man-months
- EAF → Effort Adjustment Factor
  - Derived from Cost Drivers
- E → Exponent
  - Derived from five scale drivers
    - Precedentedness
    - Development Flexibility
    - Architecture / Risk Resolution
    - Team Cohesion
    - Process Maturity

Intermediate Model: Example

- Project A is to be a **32,000 DSI semi-detached software**. It is in a mission critical area, so the **reliability** is high (RELY=high=1.15).

Then we can estimate:

- **Effort** = $1.15*3.0*(32)^{1.12}$ = 167 man-months
- **Schedule** = $2.5*(167)^{0.35}$ = 15 months
- **Productivity** = 32,000 DSI/167 MM
  = 192 DSI/MM
- **Average Staffing** = 167 MM/15 months
  = 11 FSP

Intermediate Model: Limitations

- Estimates are within **20%** of the actuals **68%** of the time
- Its effort multipliers are phase-insensitive
- It can be very tedious to use on a product with many components

Detailed COCOMO Model: How is it Different?

- **Phase-sensitive Effort Multipliers**
  Effort multipliers for the cost drivers are different depending on the software development phases
- **Module-Subsystem-System Hierarchy**
  - The software product is estimated in the three level hierarchical decomposition.
  - The fifteen cost drivers are related to module or subsystem level

Detailed COCOMO Model: When Should You Use It?

- The Detailed Model can estimate
  - **the staffing, cost, and duration of each of the development phases, subsystems, modules**

- It allows you to experiment with different development strategies, to find the plan that best suits your needs and resources

Detailed Model: Equations

- Same equations for estimations as the Intermediate Model
- Uses a very complex procedure to calculate estimation.
  - **The procedure uses the DSIs for subsystems and modules, and module level and subsystem level effort multipliers as inputs**

Detailed Model: Limitations

- Requires substantially more time and effort to calculate estimates than previous models
- Estimates are within 20% of the actuals 70% of the time

COCOMO II

- Modified for more current development
- 3 increasingly detailed cost estimation models
  - Application composition
    - Prototyping efforts (UI Issues)
    - Used in a powerful CASE environment
  - Early Design
    - Focused on Architectural design phase
  - Post-Architecture model
    - Used during implementation phase

- http://sunset.usc.edu/research/COCOMOII/index.html

Data Collection

- Regardless of the method or model used, data is needed for calibration
- Programmers need to know their own "constant adjustment factors"
  - **Goal of Personal Software Process to establish such a database**

So, what can you do?

- You
  - **Don't have a historical database**
  - **Are not an expert**
- Generate estimates using multiple models and compare based on your guesses or assumptions
  - **Similar to using the models as your personal experts in Delphi method**
  - **Candidate models:**
    - **Walston and Felix (simple and easy to use)**
    - **COCOMO 2 (complicated and detailed)**
    - **DeMarco (based on UI requirements)**
- Brooks, p. 20
  - **1/3 planning, 1/6 coding, 1/4 component tests and early system test, 1/4 system test**