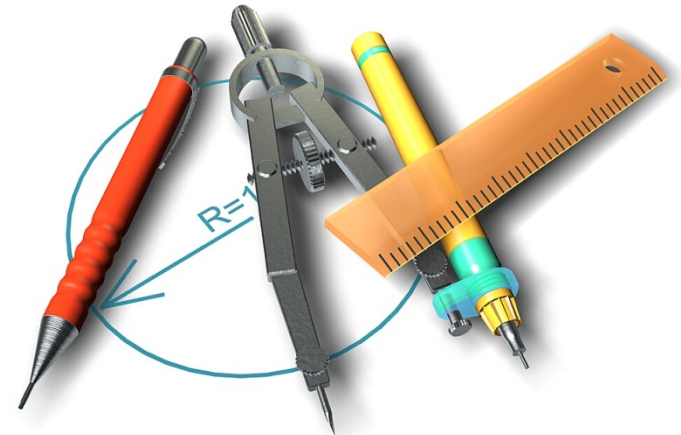


Software Cost Estimation

Nguyen Van Vu



Outline

- Software Cost Estimation
- Size Estimation
- Effort and Schedule Estimation
 - Next class



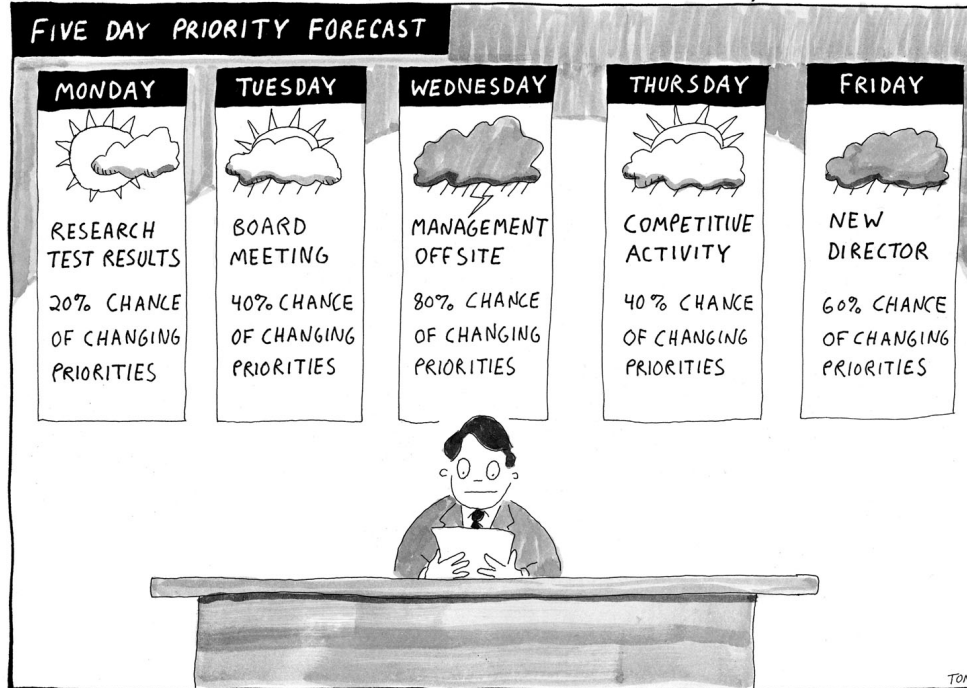
What is Software Estimation

- Software cost estimation is a process that produces effort, time, cost, etc. for developing, maintaining, and operating software systems
- Estimation is a subjective process
- The most subjective process is GUESS-TIMATE

Estimates vs. Guesstimates

BRAND CAMP

by Tom Fishburne



© 7/21/03

SKYDECKCARTOONS.COM

Why Cost Estimation is Important?

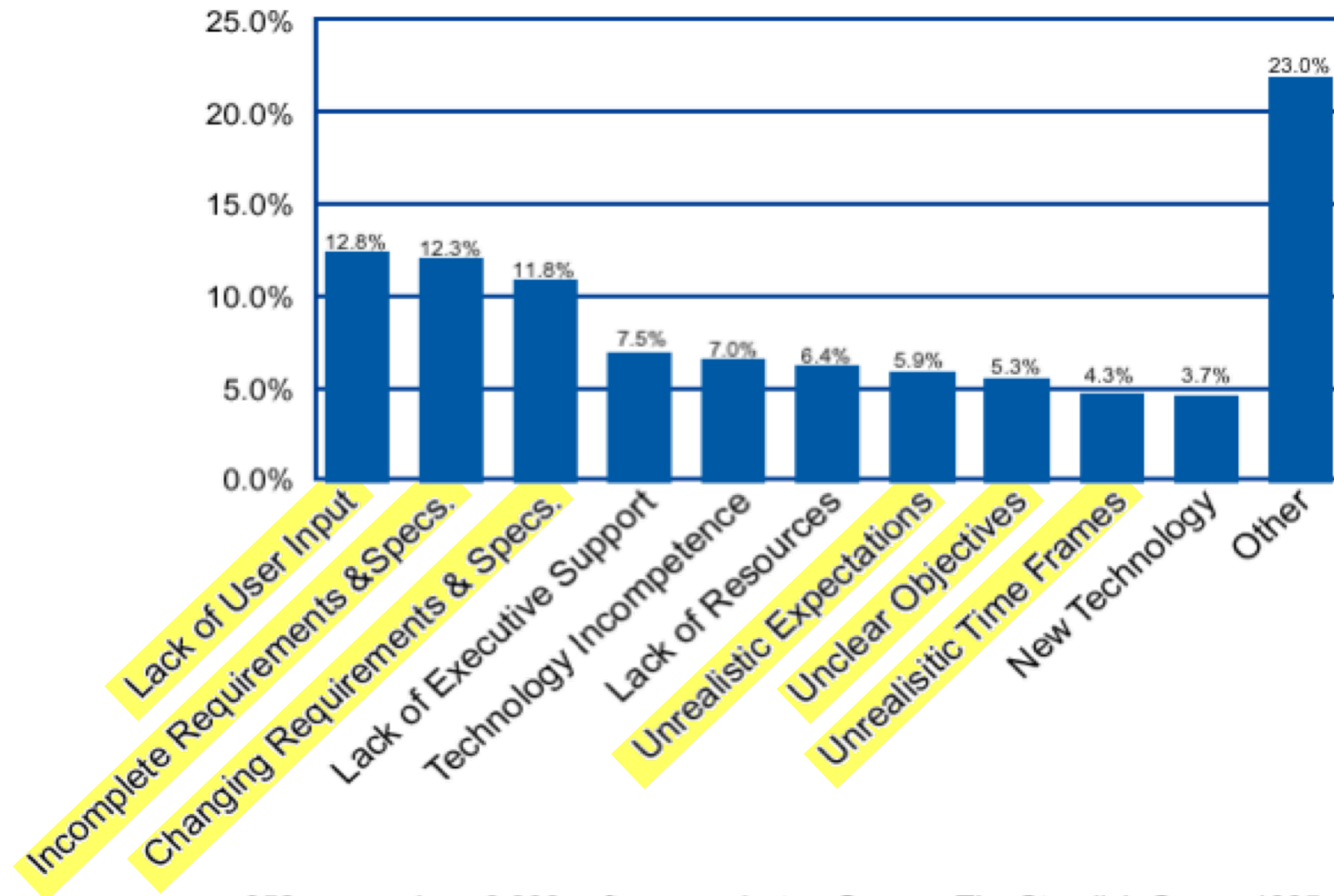
- Knowing things beforehand is always important
- For business development:
 - Investment decisions
 - Allocate resources (e.g. money, time, people) for investments
 - Project bidding dilemma
 - Overestimate → failure to get contracts → work lost → financial lost
 - Underestimate → failure to complete projects → financial lost
- For project management:
 - Project planning: effort, schedule, staff
 - Client negotiations
 - Trade-offs: change requests, risk management decisions, etc.

Win-lose Generally Becomes Lose-lose

Proposed Solution	"Winner"	Loser
Quick, Cheap, Sloppy Product	Developer & Customer	User
Lots of "bells and whistles"	Developer & User	Customer
Driving too hard a bargain	Customer & User	Developer

Actually, nobody wins in these situations

Why Software Projects Fail

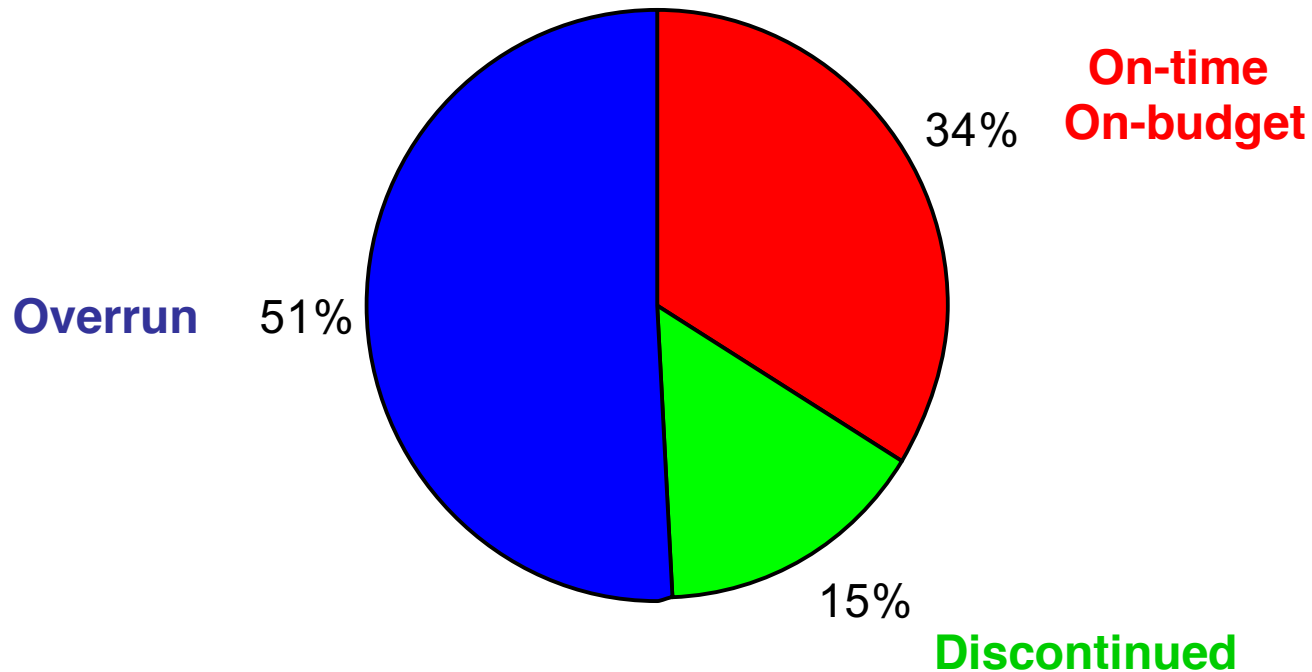


352 companies - 8,000 software projects. Source: *The Standish Group, 1995*

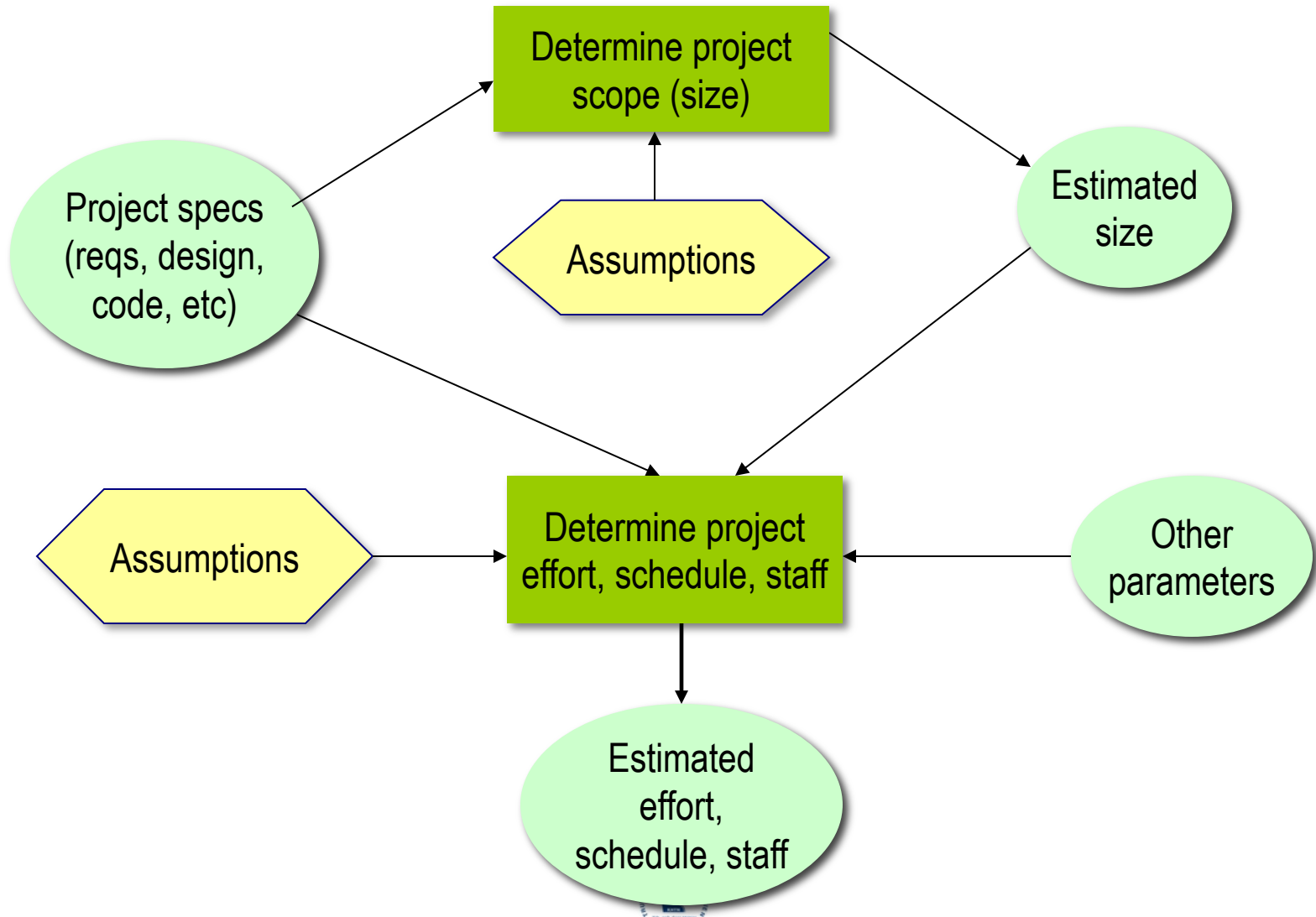


Software Engineering Is Not Well-Practiced Today

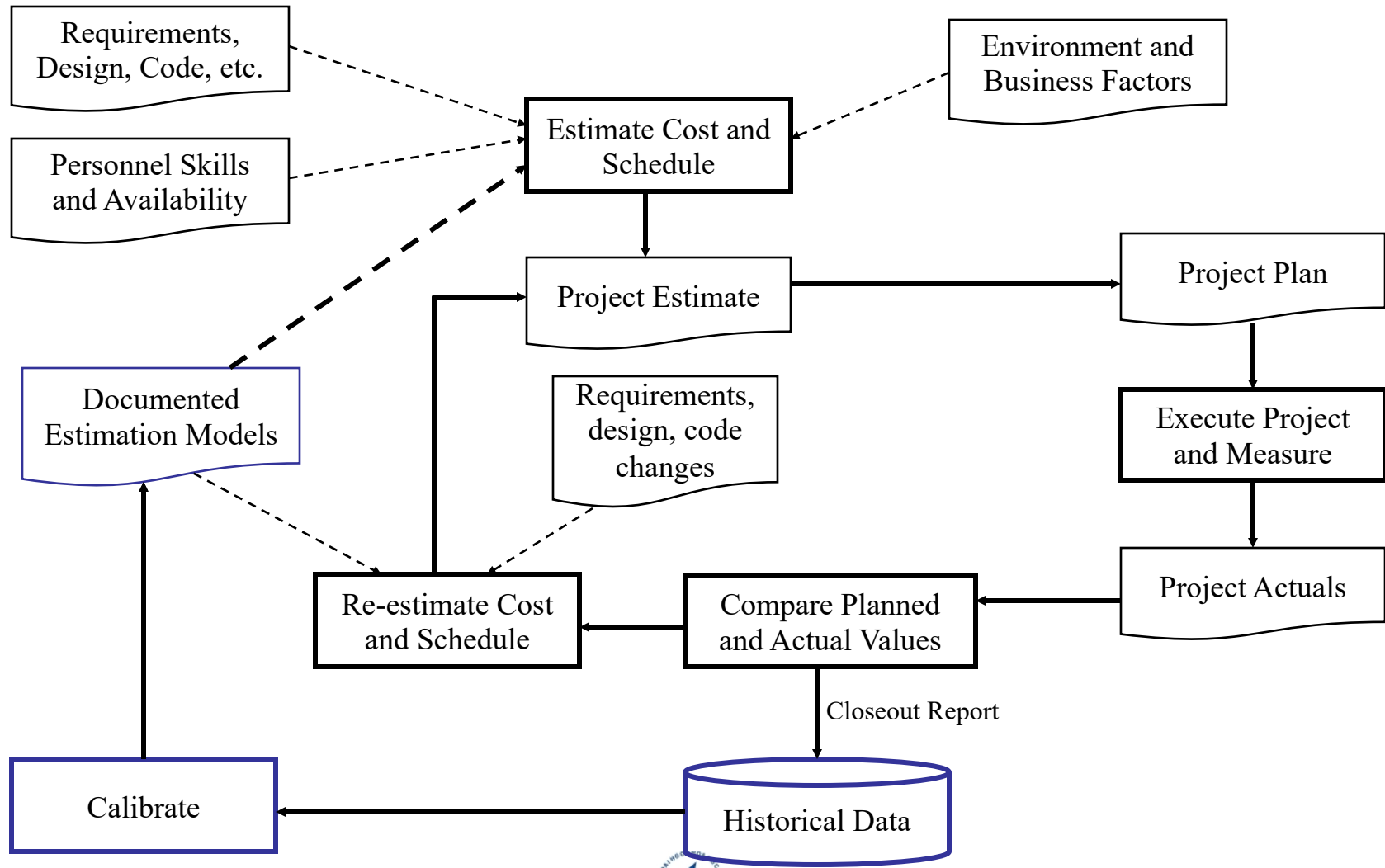
-Standish Group CHAOS Report 2003



Typical Cost Estimation Process

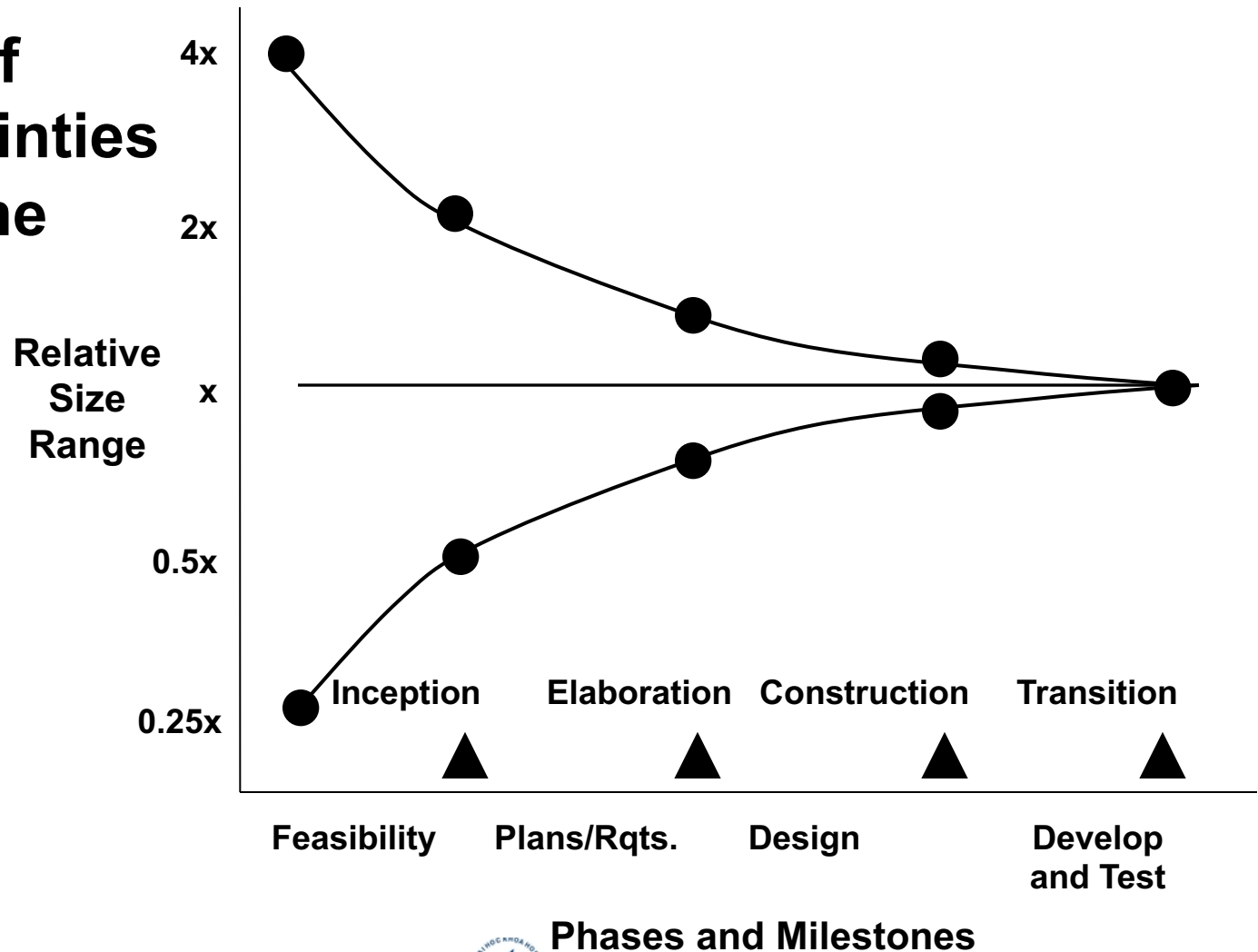


Cost Estimation Life Cycle

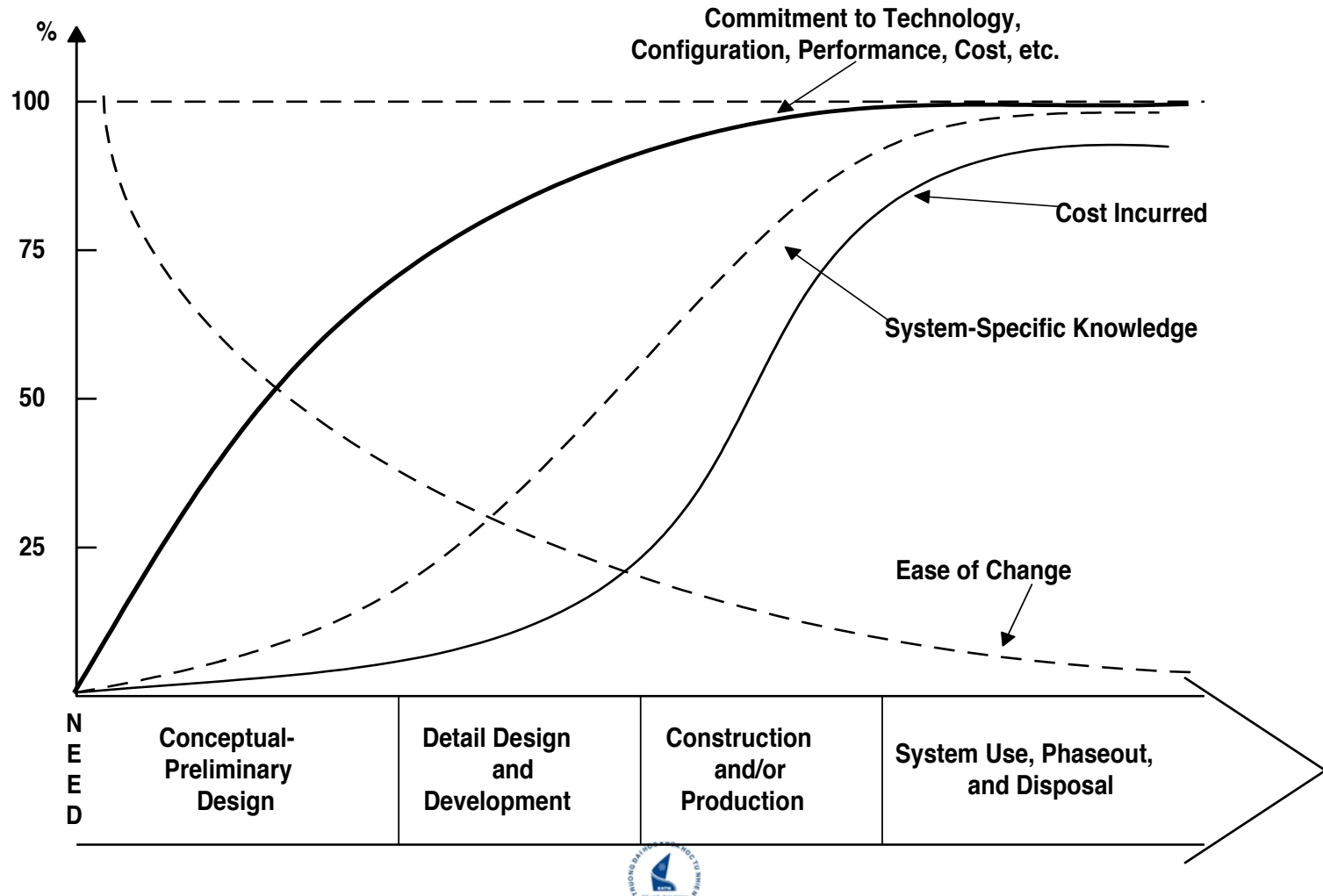


Software Estimation Accuracy

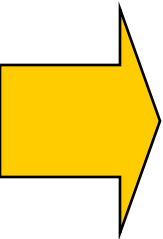
- Effect of uncertainties over time



Foreclosure of Software Options —Blanchard- Fabrycky, 1998



Outline

- 
- Software Cost Estimation
 - Size Estimation
 - Effort and Schedule Estimation
 - Next class



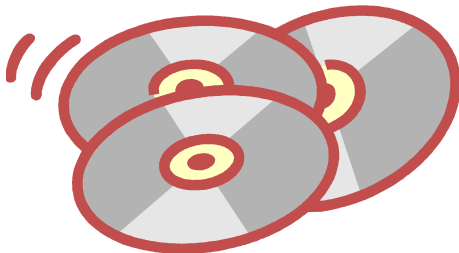
Size: Bigness/Bulk/Magnitude



Measured in gallons



Measured in usable square feet



Measured in ???

Size Estimation and Why

- Producing cost estimate directly from specs is not always possible:
 - scope of the project is unknown
 - effort and time are highly dependent on team settings
 - high “degree of guess-timate” → less objective and less accurate
- There are needs for
 - less subjective and more accurate estimates
 - size comparison among projects
 - determining productivity
 - technologically independent estimates: providing different estimates for different architectures
- Estimating size is the first and crucial step of most estimation models

Software Size is an Important Metric

- A key metric used to determine
 - software project effort and cost:

$$(effort, cost, schedule) = f(size, factors)$$

- time to develop
- staff
- quality
- productivity

When Does Sizing Add Less Value?

- Often easier to go directly to estimating effort
 - Imprecise size parameters
 - GUI builders; COTS integration
 - Familiar, similar-size applications
 - Analogy to previous effort: “yesterday’s weather”
- When size is a dependent variable
 - Time-boxing with prioritized features

Basic Methods, Strengths, and Weaknesses

(Adapted from Boehm, 1981)

Method	Strengths	Weaknesses
Pair-wise comparison	<ul style="list-style-type: none"> •Accurate assessment of relative size 	<ul style="list-style-type: none"> •Absolute size of benchmark must be known
Expert judgment	<ul style="list-style-type: none"> •Assessment of representativeness, interactions, exceptional circumstances 	<ul style="list-style-type: none"> •No better than participants •Biases, incomplete recall
Analogy	<ul style="list-style-type: none"> •Based on representative experience 	<ul style="list-style-type: none"> •Representativeness of experience
Price to win	<ul style="list-style-type: none"> •Often gets the contract 	<ul style="list-style-type: none"> •Generally produces large overruns
Top-down	<ul style="list-style-type: none"> •System level focus •Efficient 	<ul style="list-style-type: none"> •Less detailed basis •Less stable
Bottom-up	<ul style="list-style-type: none"> •More detailed basis •More stable •Fosters individual commitment 	<ul style="list-style-type: none"> •May overlook system level costs •Requires more effort


Why Do People Underestimate Size?

(Boehm, 1981)

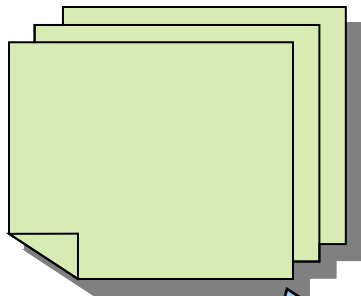
- Basically optimistic and desire to please
- Have incomplete recall of previous experience
- Generally not familiar with the entire software job

Sizing Metrics vs. Time and Degree of Detail

(Stutzke, 2005)

Process Phase	Concept	Elaboration			Construction	
Possible Measures	Subsystems Key Features	User Roles, Use Cases	Screens, Reports, Files, Application Points	Function Points	Components Objects	Source Lines of Code, Logical Statements
Primary Aids	Product Vision, Analogies	Operational Concept, Context Diagram	Specification, Feature List	Architecture, Top Level Design	Detailed Design	Code
	 <p>Increasing Time and Detail</p>					

Sizing Metrics and Methodologies



Requirements

Function Points

COSMIC Function Points

Use Case Points

Mark II Function Points

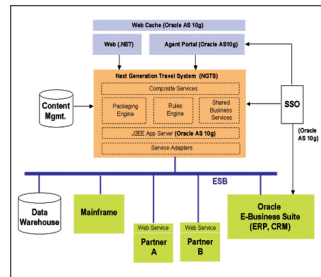


FIGURE 1 LIBGo software architecture (simplified)

Architecture and Design

CK Object Oriented Metrics

Class Points

Object Points

```
00221 Public Function GetChecksum(ByVal sentence As String) As String
00222 Dim Character As Char
00223 Dim Checksum As Integer
00224 For Each Character In sentence
00225     Select Case Character
00226     Case "$"
00227         'Ignore the dollar sign
00228     Case "*"
00229         'Stop processing before the asterisk
00230     Exit For
00231     Case Else
00232         'Is this the first value for the checksum?
00233         If Checksum = 0 Then
00234             'Yes. Set the checksum to the value
00235             Checksum = Convert.ToByte(Character)
00236         Else
00237             Checksum = Checksum Xor Convert.ToByte(Character)
00238         End If
00239     End Select
00240 Next
00241 Return Checksum.ToString("X2")
00242 End Function
```

Source Lines of Code (SLOC)

Module (file) Count

Code



Counting Artifacts

- Artifacts: requirements, inputs, outputs, classes, use cases, modules, scenarios
 - Often weighted by relative difficulty
 - Easy to count at initial stage
 - Estimates may differ based on level of detail

Major Size Estimation Metrics/Units

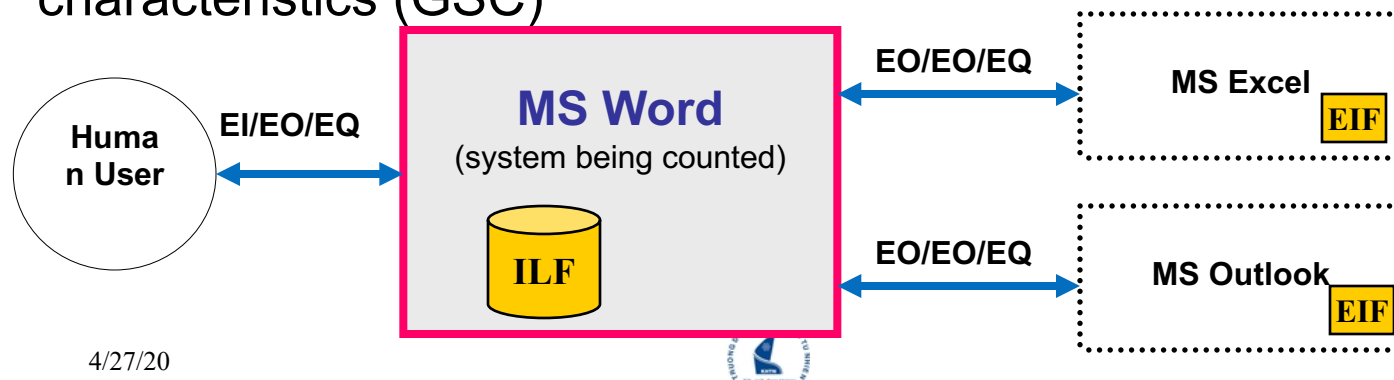
- Source Line of Code (SLOC)
- Function Points Analysis (FPA)
- Full Function Points, COSMIC FFP
- Mark II FPA
- Use Case Points (UCP)
- Object Points
- Story Points
- xyz Points
- ...

SLOC

- SLOC measures software size or length by counting the logical or physical number of source code lines
 - Physical SLOC and logical SLOC or KSLOC
- Advantages
 - Can be automated easily
 - Simple and easy to understand for customers and developers
 - Reflect the developer's view of software
 - Supported by most of cost estimation models
- Disadvantages
 - Dependent on programming languages, programming styles, and programmer skills
 - Difficult to estimate early
 - Inconsistent definitions, esp., across languages and organizations

Function Point Analysis - 1

- FPA measures the software size by quantifying functionality of the software provided to users
- Five function types
 - Data
 - Internal Logical File (ILF), External Interface File (EIF)
 - Transactions
 - External Input (EI), External Output (EO), External Query (EQ)
- Each function is weighted by its complexity
- Total FP is then adjusted using 14 general systems characteristics (GSC)



Function Point Analysis - 2

- Advantages

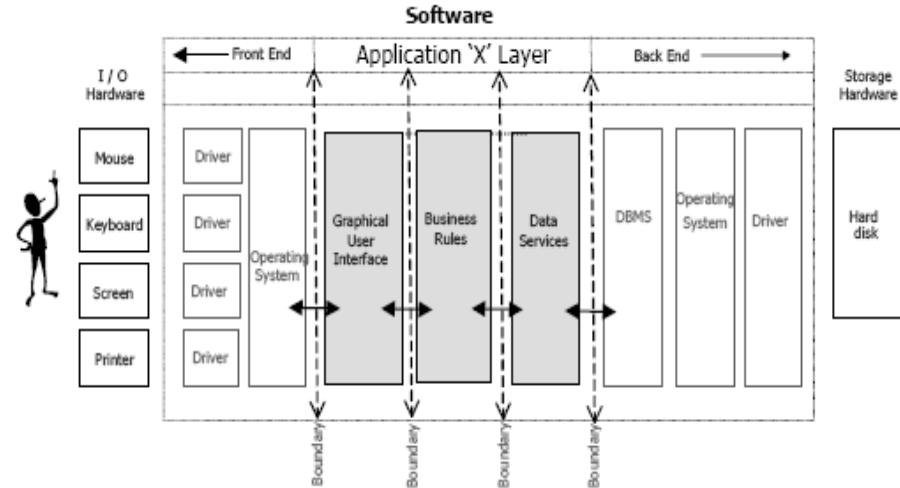
- Independent of technology and implementation
- Can be measured early in project lifecycle
- Can be applied consistently across projects and organizations

- Disadvantages

- Difficult to automate
- Require technical experience and estimation skills
- Difficult to count size of maintenance projects

COSMIC FP

- COSMIC is a functional size measurement method that measures functionality of the software thru the movement of data (exit, entry, read, write) cross logical layers
- Advantages
 - Same as FPA
 - Applicable to real-time systems
- Disadvantages
 - Difficult to automate
 - Require experience and estimation skills
 - Require detailed architecture
 - Lack of capability to measure complexity scientific systems, etc.
 - Difficult to understand for customers



Use Case Point

- UCP counts functional size of the software by measuring the complexity of use cases

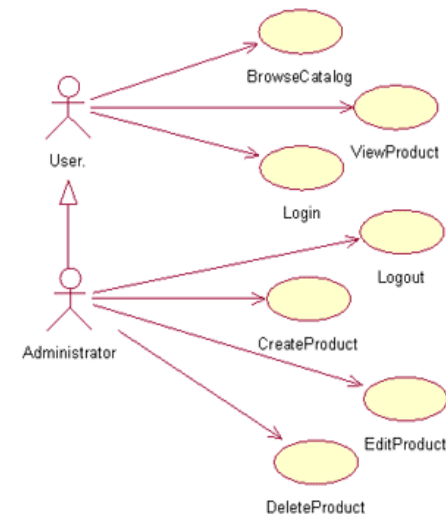
$$\text{UCP} = (\text{Actor count} + \text{Use case count}) * \text{technical and environmental factor}$$

- Advantages

- Independent of technology and implementation
- Can be measured early in project lifecycle
- Easier to count than FPA

- Disadvantages

- Lack of standardization → different between projects and orgs.
- Difficult to automate
- High variance (low accuracy)



Relationship Among Sizing Metrics

- Two broad categories of sizing metrics
 - Implementation-specific e.g. Source Lines of Code (SLOC)
 - Implementation-independent e.g. Function Points (FP)
- Need to relate the two categories
 - e.g. SLOC/FP backfire ratios

SLOC/FP Backfiring Table

(Jones, 1996): other backfire ratios up to 60% higher

Language	SLOC/UFP	Language	SLOC/UFP
Access	38	Jovial	107
Ada 83	71	Lisp	64
Ada 95	49	Machine Code	640
APL	32	Pascal	91
Assembly-Basic	320	PERL	27
Basic-ANSI	64	Prolog	64
Basic-Visual	32	Report Generator	80
C	128	2nd Generation Lang.	107
C++	55	Simulation-Default	46
Database-Default	40	3rd Generation Lang.	80
5th Generation Lang.	4	Unix Shell Scripts	107
1st Generation Lang.	320	USR_1	1
Fortran 95	71	USR_4	1
4th Generation Lang.	20	USR_5	1
High Level Lang.	64	Visual Basic 5.0	29
HTML 3.0	15	Visual C++	34
Java	53		

Outline

- Software Cost Estimation
- Size Estimation
- Effort and Schedule Estimation
 - Next class



Effort and Schedule Estimation

- Generating effort, schedule, staff estimates based on project's parameters: size, characteristics, environment settings, etc.
- Most estimation models use size as main input for effort and schedule estimation
- Models:
 - COCOMO Models
 - SLIM Model
 - PRICE-S
 - SEER-SEM
 - Delphi
 - Work Breakdown Structure (WBS)
 - Regression
 - ...

Software Cost Estimation Methods

- Cost estimation: prediction of both the person-effort and elapsed time of a project
- Best approach is a combination of methods
 - compare and iterate estimates, reconcile differences
- COCOMO - the “COConstructive COst MOdel”
 - COCOMO II is the update to Dr. Barry Boehm’s COCOMO 1981
- COCOMO is the most widely used, thoroughly documented and calibrated cost model

References

- IFPUG Counting Practice Manual v4.2
- ISO/IEC 19761:2002 COSMIC-FFP
- ISO/IEC 20926:2002 IFPUG 4.1 Unadjusted
- ISO/IEC 20968:2002 Mk II Function Point Analysis
- ISO/IEC 24570:2004 NESMA functional size measurement method version 2.1