Lecture Slides for
Managing and Leading Software Projects

Chapter 6: Estimation Techniques

developed by

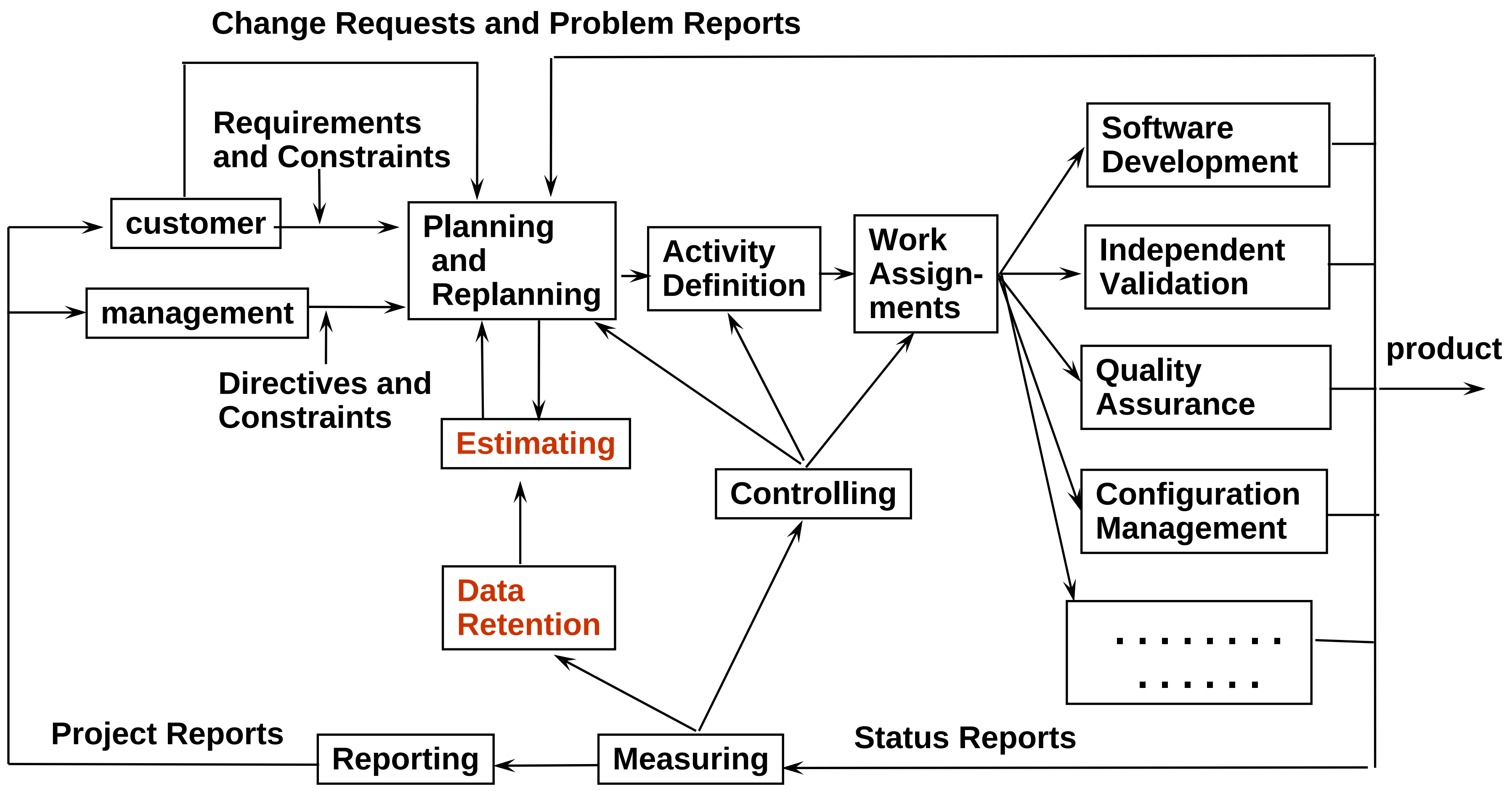
Richard E. (Dick) Fairley, Ph.D.

to accompany the text

Managing and Leading Software Projects

published by Wiley, 2009

A Workflow Model with Emphasis on Estimation for Software Projects



Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Software Size
- Pragmatic Estimation Techniques
- Parametric Estimation Models
  - Theory-Based Estimation Models
  - Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

Additional Information

- The four sets of standards and guidelines for managing project presented in the text; namely the CMMI-DEV-v1.2 process framework, the ISO/IEEE standard 12207, IEEE standard 1058, and the PMI Body of Knowledge address estimation issues to varying degrees. Aspects of estimation in these documents are presented in Appendix 6A to Chapter 6.
- Terms used in Chapter 6 and throughout the text are defined in Appendix A to the text.
- Presentation slides for this chapter and other supporting material are available at the URL listed in the Preface to the textbook.

Objectives for Chapter 6

- After reading this chapter and completing the exercises you should understand:
  - The role of estimation in the workflow model for software projects
  - Three fundamental principles of estimation
  - Size measures and size measurement
  - How to develop a size measure
  - Some pragmatic, theory-based, and regression-based estimation techniques
  - How to develop, calibrate, and evaluate the acceptability of regression-based estimation models
  - Capabilities of estimation tools
  - An estimation procedure
  - A format for documenting estimates

The Goal of Estimation

- The goal of estimation is to determine a set of parameters that provide a high level of confidence you will be able to delivering an acceptable product within the bounds of the project constraints.
- The parameters and constraints to be considered are:
    - product features,
    - quality attributes,
    - effort,
    - other resources,
    - schedule,
    - budget,
    - technology, and
    - a basis of estimation.

some of these may be specified as constraints and others are to be estimated.

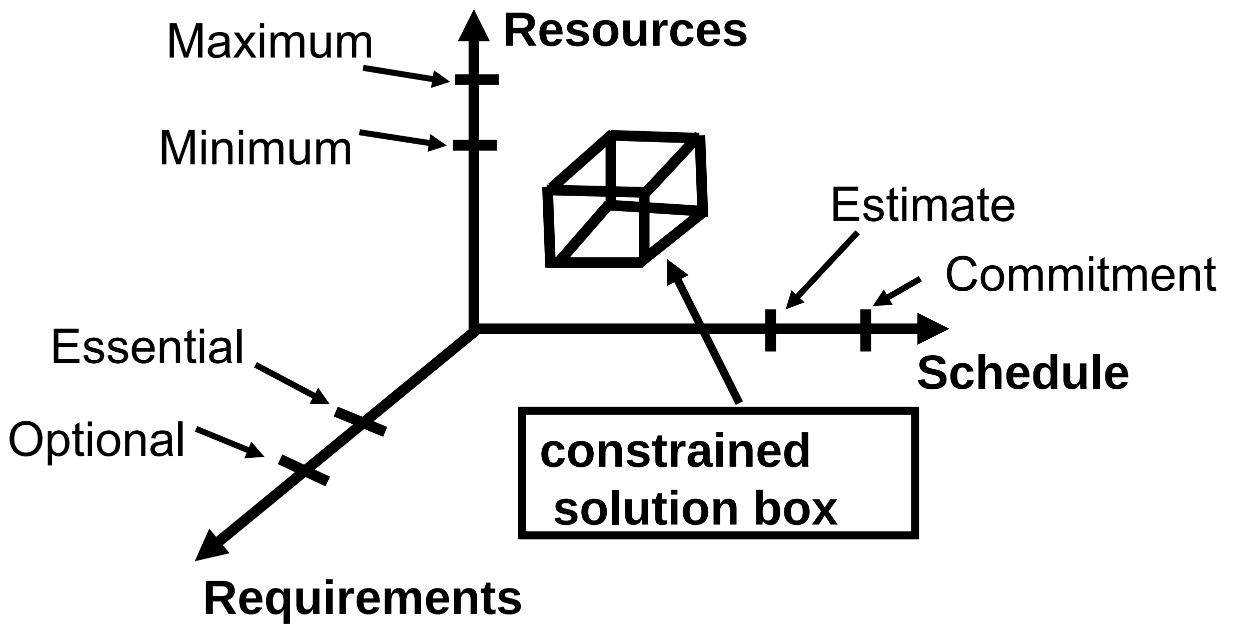
Section 5.1.1 of Annotated IEEE Standard 1058*

5.1.1 Estimation Plan

- What is the plan for making initial and on-going estimates?
- What are the initial estimates:
    - What are the details of the project cost, schedule, staff requirements, and other resources?
    - What methods, tools, and techniques were used to make the estimates?
    - What historical information was used?
    - What is the estimator's level of confidence in the estimate?
- How will periodic re-estimates be made of cost, schedule, staffing, and other resources required to complete the project?
- How frequently will re-estimation be done?
- What is the plan for re-estimating when requirements or other project conditions change?

* see the URL listed in the preface of the text for the annotated version of IEEE Standard 1058

Three Fundamental Variables of Estimation and the Constrained Solution Box



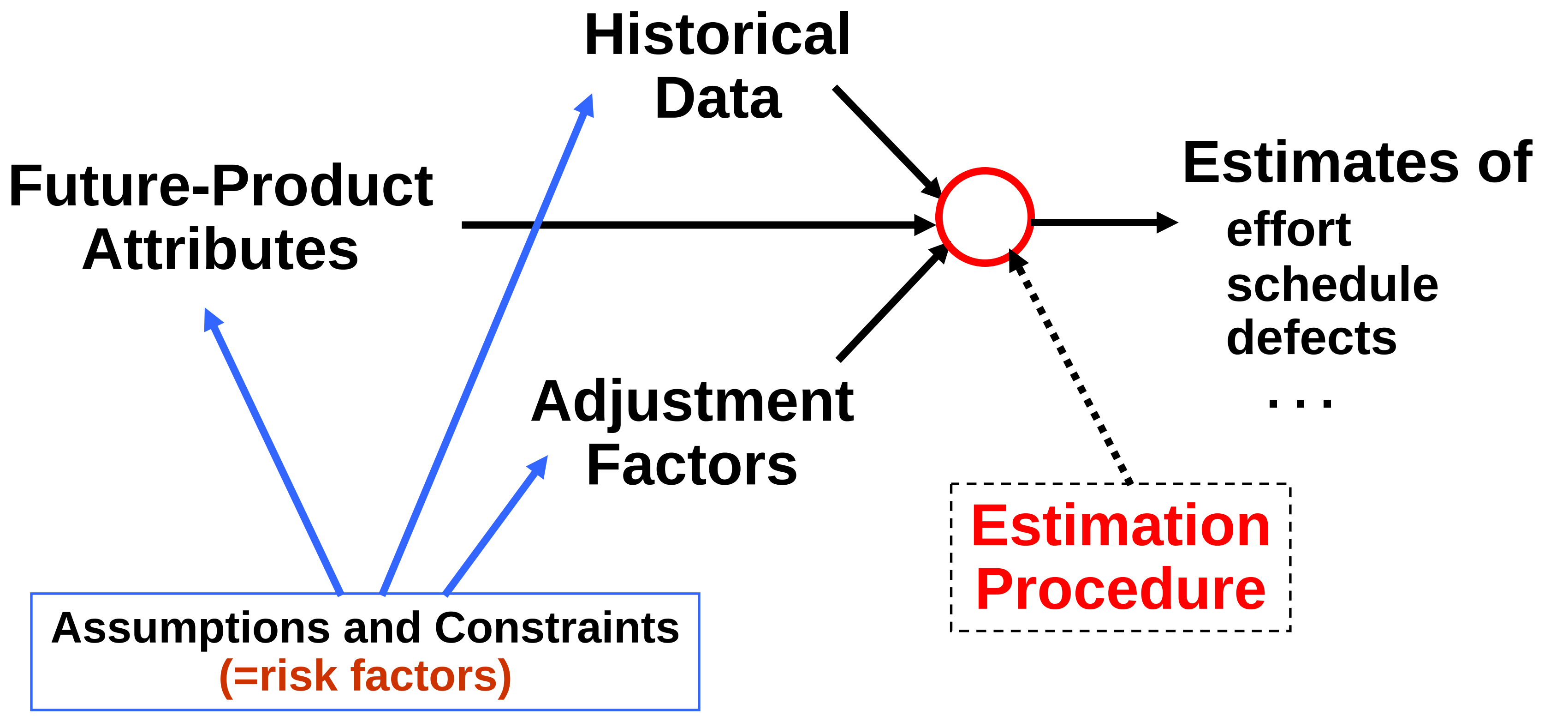
NOTE

- A rigid, fixed value for one of the 3 Fundamental Variables reduces the solution box to a two dimensional solution plane
- Rigid, fixed values for two of the 3 Fundamental Variables reduces the solution box to a solution line
- Rigid, fixed value for all 3 of the Fundamental Variables fixes the solution point
  - o an leaves no flexibility for tradeoffs among
    - resources,
    - schedule, and
    - requirements
  - o which usually a recipe for disaster

The Fundamental Principle of Estimation

- An estimate is a projection from past to future, suitable adjusted to account for differences between past and future
  - o the past is captured by historical data
  - o the future is captured in the requirements for the software to be developed
    - and the project constraints
  - o the differences are adjustment factors to account for:
    - different customer
    - different development team
    - different tools
    - different application
    - and so forth

Elements of Estimation



Estimating Cost

- Estimates of cost are typically made by estimating effort
  - and multiplying by the loaded cost* per unit of effort
  - and by adding other costs such as equipment, travel, and infrastructure support
- An example:
  - an estimated 50 units of effort at $2500 per unit will cost $125,000
  - if an additional $100,000 is needed for other items the total cost of the project is estimated to be $225,000

* loaded cost is the cost to the organization for units of effort; it includes factors, in addition to salary, such as health insurance, retirement, and vacation time

Estimation Principle #1

Estimation Principle #1:

A project estimate is a projection from past experiences to the future, adjusted to account for differences between past and future.

Three things are apparent from this principle:

1. you must have some past experiences to draw upon
   - known as the basis of estimation
2. you must know something about the future
   - requirements for the system or product you will develop or modify; constraints on the project
3. you must make adjustments to account for the differences between past and future
   - known as the adjustment factors

Estimation Principle #2

Estimation principle #2:

All estimates are based on a set of assumptions that must be realized and a set of constraints that must be satisfied.

- Said differently, your estimate will be invalid if you fail to satisfy the assumptions made in preparing the estimate
  - o the project will fail to meet its goals if the constraints are violated.

Assumptions and Constraints

- An *assumption* is a statement that is taken to be true without verifying, or being able to verify, the truth of the statement
    - for example, it might be assumed that the productivity factor on the next project will be 500 delivered source lines of code per staff-month (500 DSLOC/SM)
- A *constraint* is an externally imposed condition that must be observed
    - for example, the project might be constrained to 5 people for 6 months

assumptions made and constraints imposed are major risk factors when making estimates

Estimation Principle #3

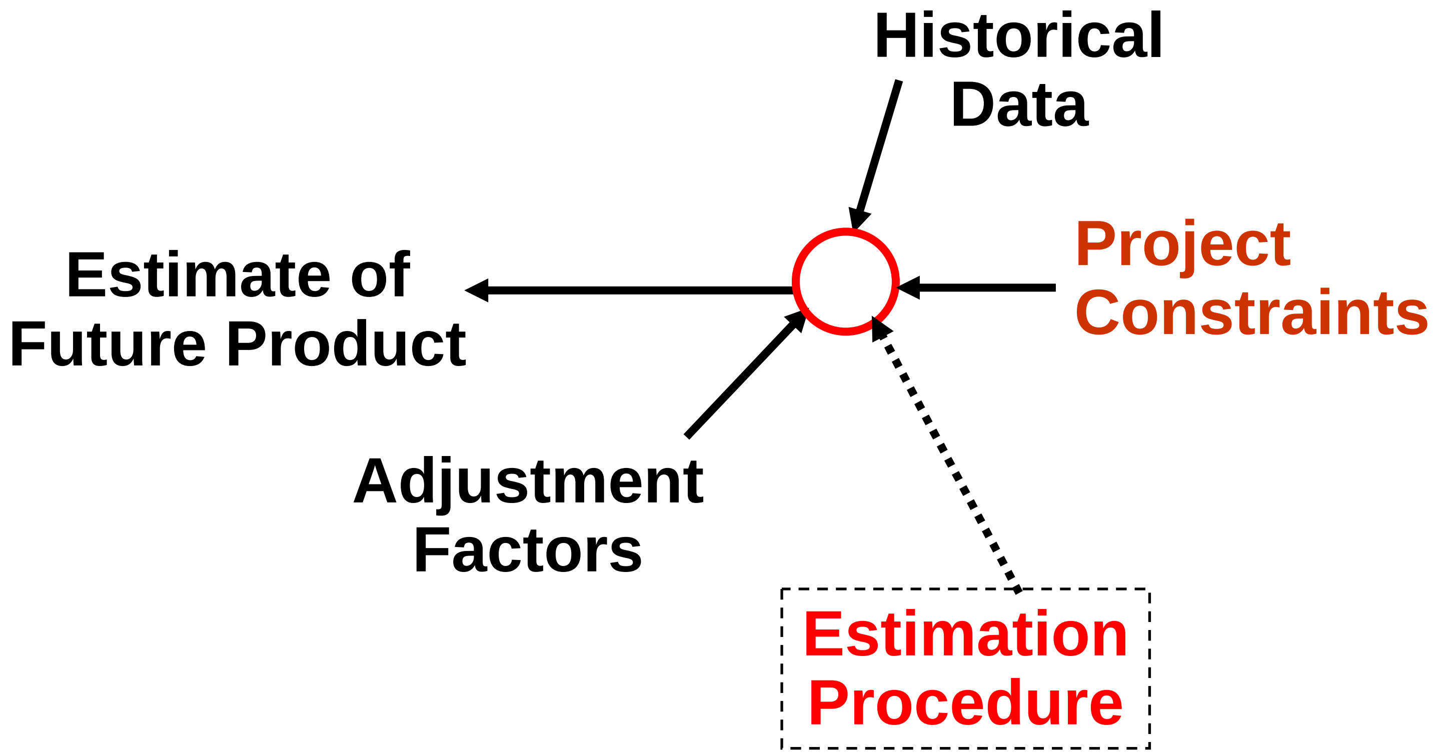
Estimation principle #3:

Projects must be re-estimated periodically as understanding grows and aperiodically as project parameters change.

- This principle is a corollary to Principle #2.
    - As your project evolves, your understanding of the product under development, the assumptions you have made, and the impact of the constraints will become clear (clearer).

Estimating with Constraints (Designing to Cost and/or Schedule)

- Given a set of project constraints, what product can we build?



An Example

- Constraints: 5 people and 6 months,

  available effort is:

  5 x 6 = 30 staff-months

- Suppose similar past projects have had an average productivity level of 500 delivered source lines of code per staff-month (DSLOC/SM)

  then, *if this project is like the typical past project*, it will produce:

  500 x 30 = 15,000 DSLOC

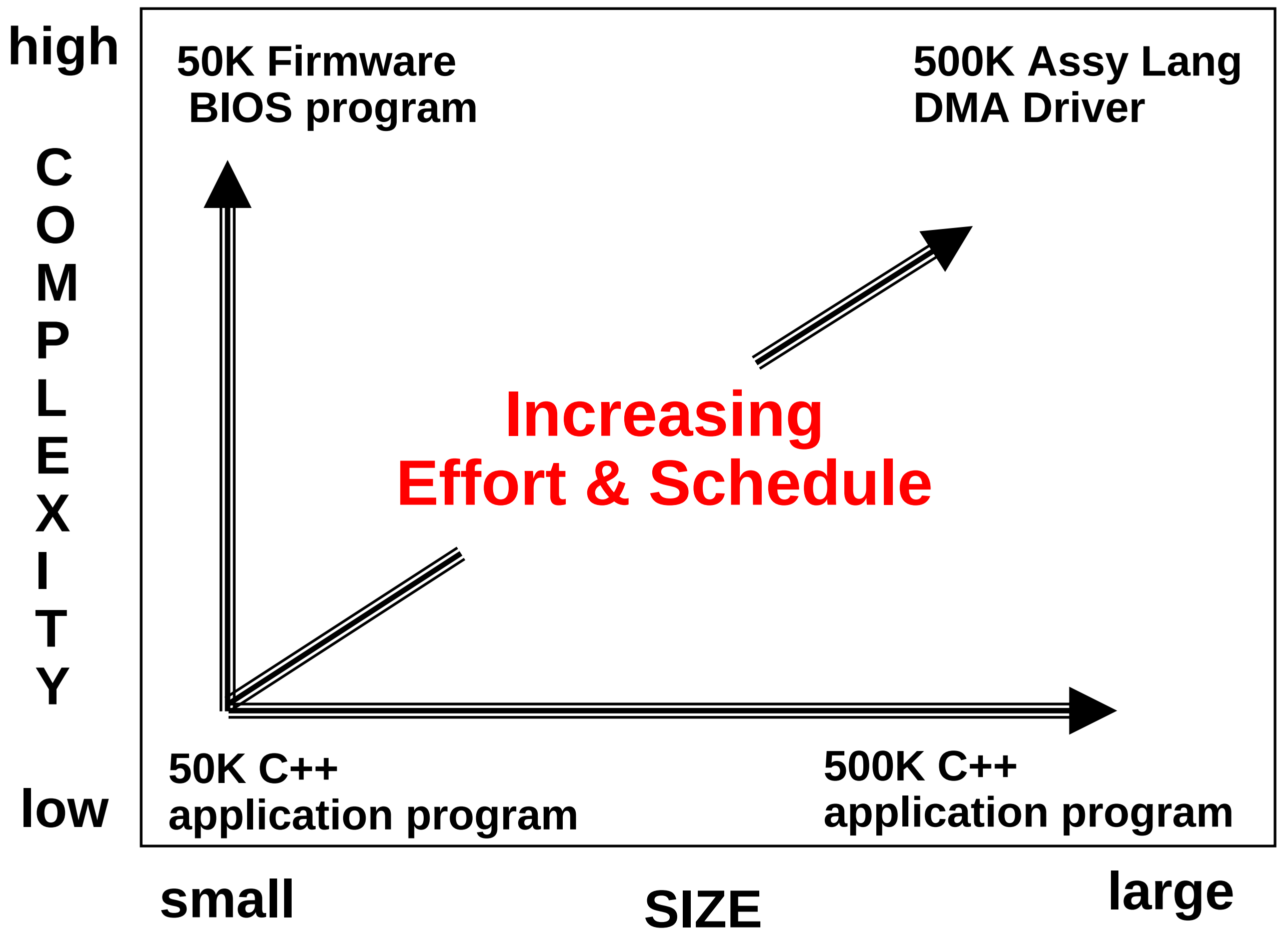
Questions:

1. what if the future project is not like the typical past project?
2. what if we think the future product will be about 30,000 LOC?

The Role of Adjustment Factors

- The future project may differ from past projects in ways that will make the developers more or less productive than the average of past projects
- Typical adjustment factors include:
  - stability of the requirements
  - relationship with the customer
  - ability of the software developers
  - schedule constraint
  - familiarity with the problem domain
  - familiarity with the development platform and software tools
- Each of these factors, and other factors, may make the software team more productive or less productive than in the past

An Adjustment Factor: Product Complexity


high
COMPLEXITY
low
50K Firmware BIOS program
500K Assy Lang DMA Driver
Increasing Effort & Schedule
50K C++ application program
500K C++ application program
small
SIZE
large

complex products require more time and effort than simple products of the same size

A Note

Q: What if there is no relevant historical data for similar past projects?

- because you don’t have any
- because the next project is different than past projects for which you have data
  - for example, switching from C++ to Java

A:

- try to find some analogies from other sources
  - e.g., what have others experienced in switching from C++ to Java?
- use evolutionary development to “feel your way”
  - until you acquire some data on which to base estimates

Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Software Size
- Pragmatic Estimation Techniques
- Parametric Estimation Models
  - Theory-Based Estimation Models
  - Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

Some Ways to Measure Software Size

- Lines of code
- Function points
- Number of use cases
- UML classes and relationships
- Windows, menus, buttons
- Values, sensors, alarms
- Interrupts, priority levels, responses

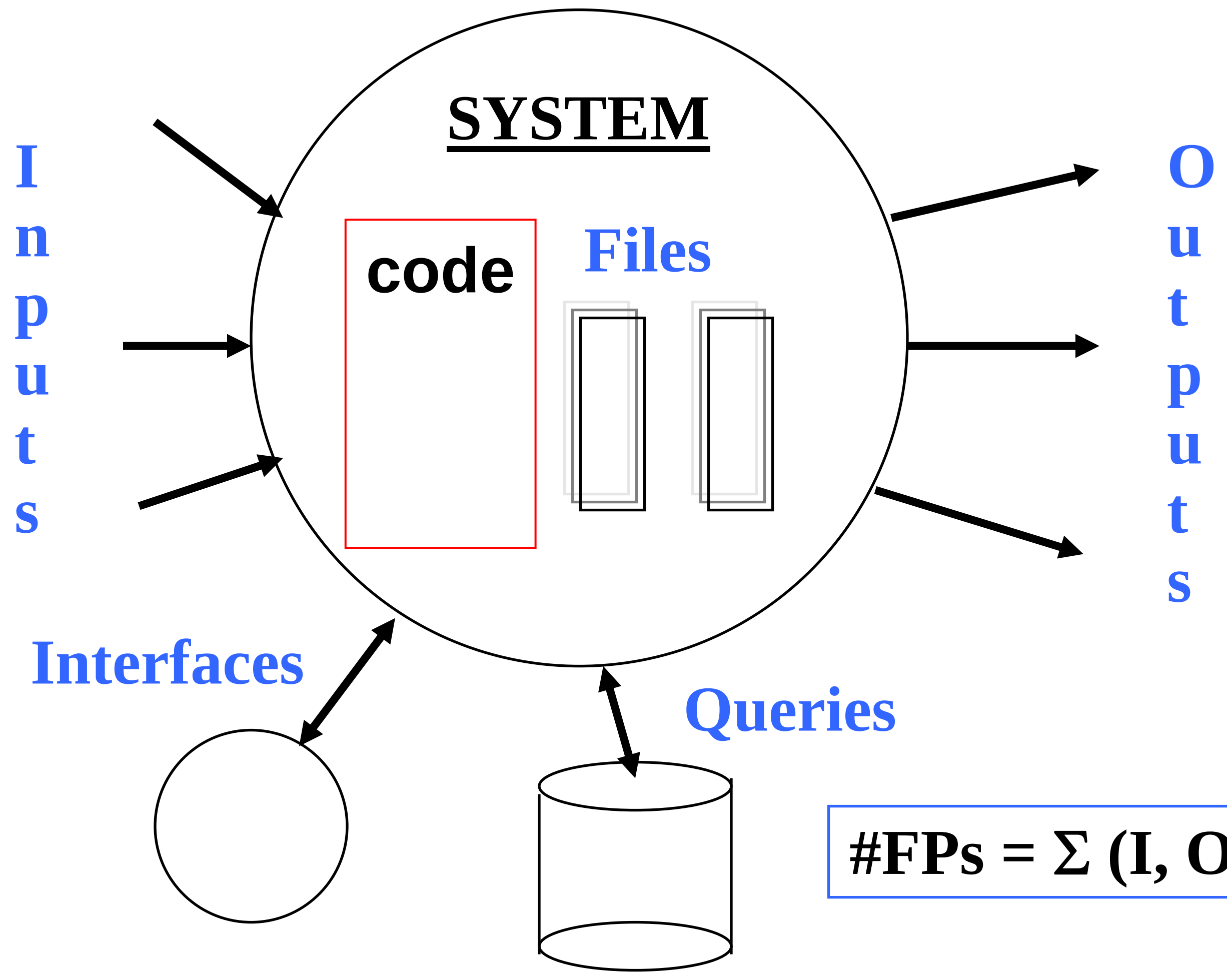
Reasons to Estimate Product Size

1. Size has a stronger causal relationship to project attributes such as effort and schedule than do other product attributes
2. Size can be measured more objectively than other product attributes
3. Some size measures can be estimated more accurately from the requirements than can other product attributes
4. Data relating size, effort, schedule, and other project attributes can be collected from completed projects and stored in a database to provide a historical basis of estimation for future projects

Problems with using Lines of Code to Measure Software Size

- It is difficult to estimate lines of code early in a project; it is difficult to relate changes to the requirements to changes in estimated lines of code
- Calculating productivity as lines of code generated per programmer-month may encourage programmers to write lots of poor quality lines of code rather than fewer lines of high quality code
- Modern development methods such as model-driven development, object-based programming, reuse of library components, and use of open source components make the relationship between lines of code and project attributes less relevant and less precise than in the past

An Alternative: The Function Point Size Measure



$$\boxed{\#FPs = \Sigma\ (I, O, F, Q, In)}$$

A Function Point Example

An estimate is derived from externally observable properties of the software to be developed or modified

- i.e., requirements and design

Calculation of function points:

number of input FPs:	25
number of output FPs :	35
number of file FPs :	20
number of inquiry FPs :	15
number of interface FPs :	5
Total Function Points:	100

Adjustment factor: 1.2

Adjusted FPs: 120

Function Point Estimation

- Suppose we estimate 120 adjusted function points based on the requirements
- Also suppose our past experience indicates that we can build this kind and size of system at a rate of 10 function points per staff-month (10 FP/SM)
  - o we will need 120 / 10 = 12 staff-months
    - 2 people for 6 months?
    - 3 people for 4 months?
    - 4 people for 3 months?
    - but not 12 people for 1 month
    - and probably not 1 person for 12 months

1. How do we determine the 10 FP/SM productivity factor?
2. What if this system is not like our past systems?

Conversion of Function Points to Lines of Code

- The relationship between function points and lines of code is language dependent:
- For example:
  - MDD tools: 6 LOC/FP
  - Java: 50 LOC/FP
  - C++: 75 LOC/FP
  - Visual Basic: 100 LOC/FP
  - C: 125 LOC/FP
  - Assembly Language: 300 LOC/FP

Note: the conversion factor is application and context dependent; it should be developed locally
Q: how would you develop a conversion factor?
Q: why would you develop a conversion factor?

Lines of Code and Productivity Ratios for the Example

- Using the conversion factors on the previous slide for 120 FPs:
  - MDD: 6 x 120 = 720 LOC
  - C++ : 75 x 120 = 8500 LOC
  - Assembly Language: 300 x 120 = 36000 LOC
- Productivity leverage factors:
  - AL / AG = 36000 / 720 = 50:1
    - Application Generator to assembly language
  - AL / C++ = 8500 / 720 = 12.5:1
    - C++ to assembly language

External Size Measures

- Function points is an example of an external size measure (ESM)
  - factors in the environment of the software to be written are counted
  - these factors are applied to past projects to determine conversion factors from ESM to factors of interest
    - e.g., ESM/SM (SM: staff-month)
  - the conversion factors are used along with the ESM count for the future system to develop estimates for attributes of interest
    - i.e., SM = ESM / (ESM/SM)
    - e.g., SM = 120 / 10 = 12 staff-months

Examples of External Size Measures

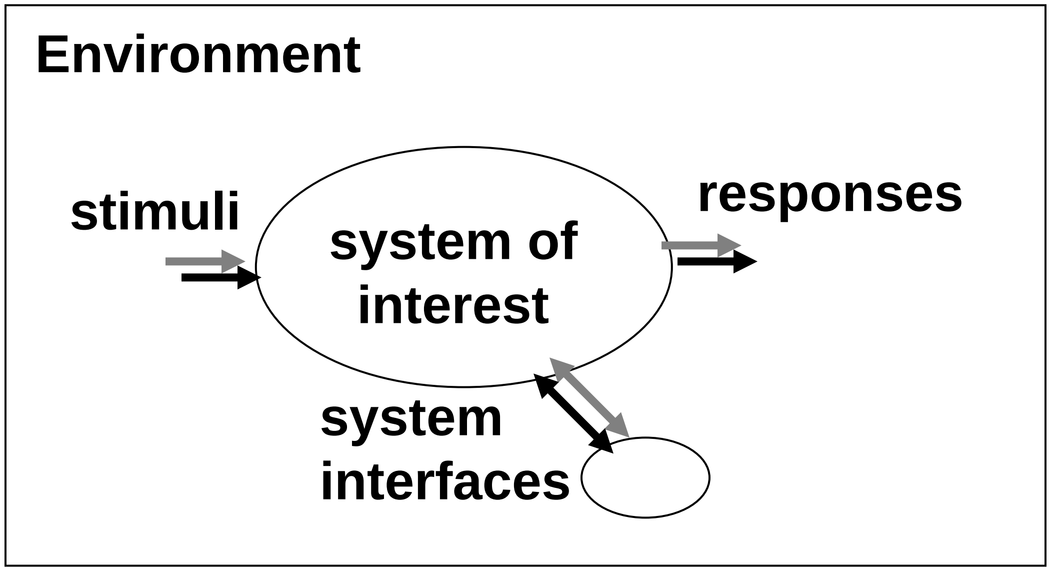
Type of system	**ESM factors counted**
Data processing	Inputs, outputs, interfaces, queries, files
Process control	Sensors, valves, actuators
Embedded systems	Interrupts, signals, priority levels
User interfaces	Windows, menus, items per menu
Object oriented	Classes, associations, methods

The ESM Conjecture

It is always possible to find an External Size Measure that can be used, along with historical data and adjustment factors, to develop estimates for project attributes of interest

Developing External Size Measures

- To estimate effort, schedule, defect levels, etc., use factors in the environment of the software to be implemented



An Example

- Suppose we estimate a new User Interface will have
  - 5 user windows,
  - 6 menus,
  - 30 items on the 6 menus, and
  - 7 buttons.
- Also suppose we have developed the following relationship from past project data:
- Effort = 4.2 * #screens + 3 * # menus + 1.2 * #items + 0.5 * #buttons
- Then the UI will require

  4.2*5 + 3*6 + 30*1.2 + 0.5*7 = 74.5 staff-hours of effort

  i.e., 1 person, 2 FTE weeks or 2 people, 1 FTE week

- We may include some adjustment factors to account for factors that make the future project different from past projects of the same size

Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Product Size
- Pragmatic Estimation Techniques
- Theory-Based Estimation Models
- Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

Pragmatic Estimation Techniques

- Pragmatic estimation techniques include:
    - WBS – CPM – PERT
    - Analogy
    - Rule of Thumb
    - Expert Judgment
    - Delphi Estimation

WBS, CPM, PERT

- The WBS, CPM, PERT approach to estimation is presented in Chapter 5
  - o estimate the effort for each task in the WBS
    - aggregate the effort estimates for the tasks
  - o specify the predecessor tasks and successor tasks for each task
  - o determine the critical path (or paths) to find the optimal schedule
  - o use the PERT approach to estimate the probability of achieving various schedule dates
  - o adjust the schedule and/or resources to achieve an acceptable staffing profile

Estimation by Analogy (1)

- Analogy is one of the most widely used estimation techniques
- Simple analogy:

  based on the requirements, it appears that a similar job took 5 people 6 months to complete

  a few of the requirements are different, so we will plan the project for 5 people, 8 months

Questions:

1. what are the future product attributes?
2. what is the historical data?
3. what are the adjustment factors?

Estimation by Analogy (2)

- Estimation by analogy can be quite sophisticated:
  - describe the attributes of your project and product
  - search for similar projects in your historical database of past projects
  - use the similar projects as the basis of estimation
  - make adjustments as necessary

Estimation by Rule of Thumb (1)

- Rule of thumb can be based on industry averages or local data for your kind of projects
- A typical productivity rule of thumb
  - o our productivity is typically 500 LOC/SM
- Typical quality rules of thumb
  - o our typical defect rate during development is typically 20 defect per KLOC
  - o our defect capture rate is typically 90%
- A typical schedule rule of thumb
  - o it generally takes about one month to do final system testing

Estimation by Rule of Thumb (2)

- Suppose we estimation our next system will contain 50,000 LOC
  - o perhaps determined by analogy
- If our productivity ROT is 500 LOC/SM, we will need 50,000 / 500

  100 SM of effort
- Using the square root ROT, we might plan the project as:

  10 people for 10 months
- If our defect ROT is 20 defects per KLOC, with a 90% capture rate, we should expect to inject 20 x 50 = 1000 defects, with

  900 detected prior to product release

  100 reported by users in the first 12 months of use

Questions:

1. what are the future product attributes?
2. what is the historical data?
3. what are the adjustment factors?

Estimation by Rule of Thumb (3)

- Some questions to consider:
  1. what scope of effort is included in the productivity ROT (500 LOC/SM)?
  2. what kind of work hours are included in the productivity ROT?
  3. is the productivity ROT for FTEs?
  4. what types of defects are considered in the defect ROT (20 per 1000 LOC)?
  5. what ROT should we use for allocation of the effort and schedule?

     allocation:

     % of effort and time to be allocated for requirements, design, coding, testing, CM, QA, project management, etc

FTE: Full-Time Equivalent software developers

Estimation by Rule of Thumb (4)

Suppose your ROT for effort and schedule allocation is:

Activity	percent of effort	percent of schedule
requirements	10%	20%
design	10%	20%
implementation	40%	30%
integration & system testing	30%	20%
acceptance & delivery	10%	10%

NOTE: the amounts of effort and schedule for the various work activities may be intermixed in an iterative manner

Estimation by Rule of Thumb (5)

- For our 10 person, 10 month project we should plan:
  - requirements: 10 SM / 2 months (5 people)
  - design: 10 SM / 2 months (5 people)
  - implementation: 40 SM / 3 months (~13 people)
  - integration & system testing: 30 SM / 2 months (15 people)
  - acceptance & delivery: 10 SM / 1 month (1 person)

NOTE:

- 10 people is the average FTE staffing level
- which can be used to estimate cost

Estimation by Expert Judgment

- Expert judgment is typically
  - combined with “estimation by parts”
- Estimation by parts:
  - the high-level product structure (i.e., the ADV) is determined
    - requirements and interfaces are allocated to each element of the product
  - experts are asked to estimate effort and schedule for each part in which they are experts
  - the individual estimates are combined to produce an overall estimate
- Two cautions:
  1. experts will estimate what it would take them to do the job
  2. you must add in effort estimates for other tasks, such as: project management, integration and test, documentation, CM, QA, V&V, etc
     - perhaps another 50%

Expert Judgment / Delphi Estimation

1. A coordinator gives each expert the information on which to base the estimate
2. Experts work alone and submit their estimates and their rationales to the coordinator
3. The coordinator prepares an anonymous report that contains the estimates and rationales of each estimator and gives the report to each estimator and asks each to submit a second estimate
4. The procedure continues until the estimates stabilize
   - usually after 3 or 4 rounds
   - If there is small disparity in the stabilized estimates, they can be used as the range of estimates
   - If there is wide disparity in the stabilized estimates, the estimators meet to discuss and resolve their disagreements

Estimation for Reuse

- Reuse of existing code may require:
  - effort to locate candidate code
  - effort to evaluate candidate code
  - effort to modify and test chosen code
  - effort to integrate chosen code
- The COCOMO II estimation model (later) includes an equation for estimating the cost in effort to reuse existing software

Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Software Size
- Pragmatic Estimation Techniques
- Parametric Estimation Models
  - Theory-Based Estimation Models
  - Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

Parametric Estimation Models

- Theory-based and regression-based estimation models are parametric models
  - o estimates are computed using equations that have some input parameters, such as:

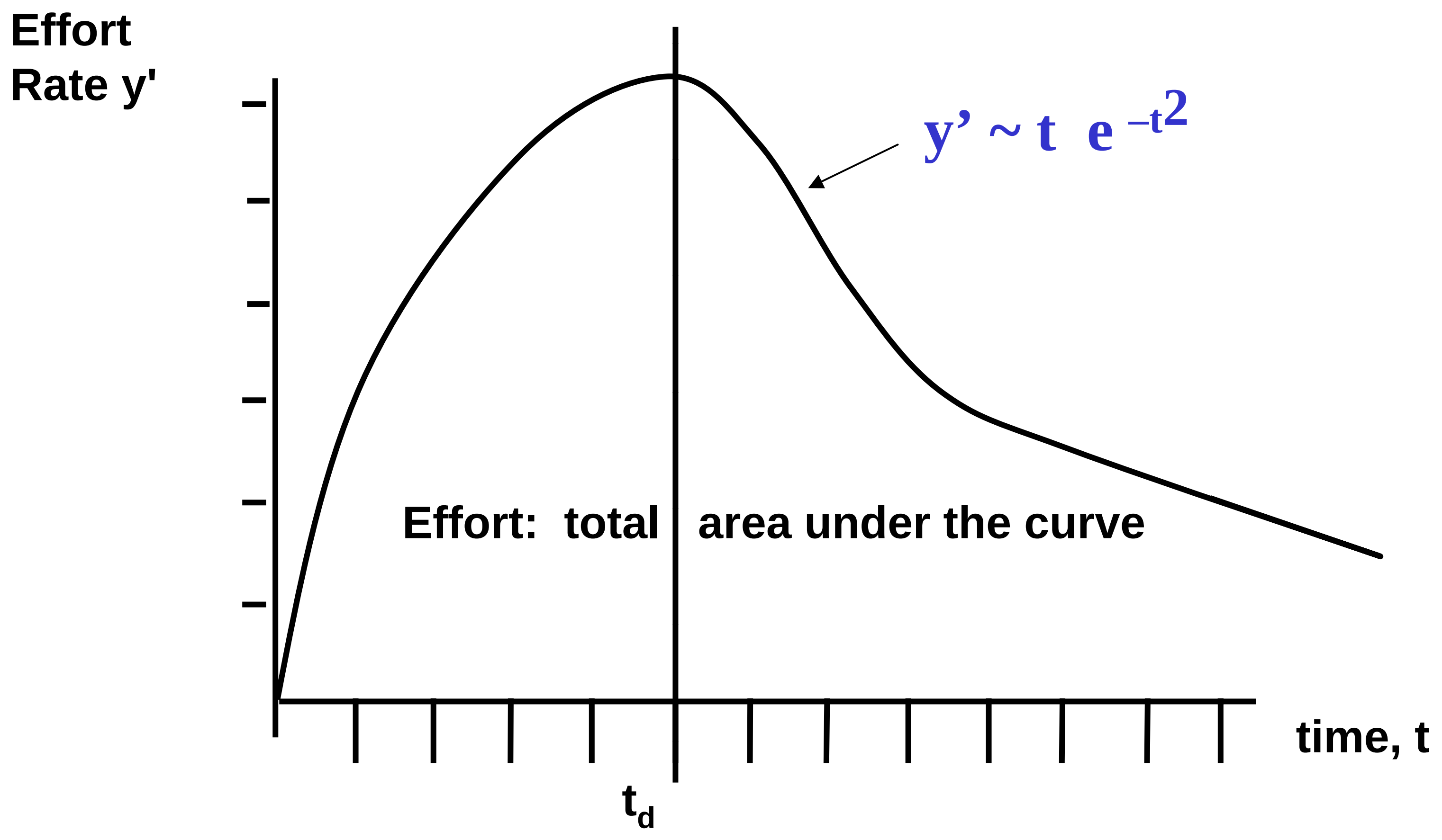
    Effort = f(p1, p2, . . . pn)
- For example:
  - o $\text{Effort} = a * \text{Size}^{b} * \text{Time}^{c}$
- where:
  - o Size is estimated product size
  - o Effort is in staff-months
  - o Time is schedule in months
  - o a, b & c are constants
    - determined by local circumstances

parametric estimation models use Size as the primary input variable

SLIM: A Theory-based Estimation Model

- SLIM: Software Lifecycle Management
- SLIM is a commercial estimation tool licensed by the QSM Company
- The estimation model is based on two equations in two unknowns (effort and time)
  - o simultaneous solutions to the equations provide estimates as combinations of effort and time
  - o impossible solutions are indicated
    - e.g. 60 persons for 1 month is an impossible estimate

The Theory: The Norden-Rayleigh Effort Equation



The Original SLIM Equations (in simplified form)

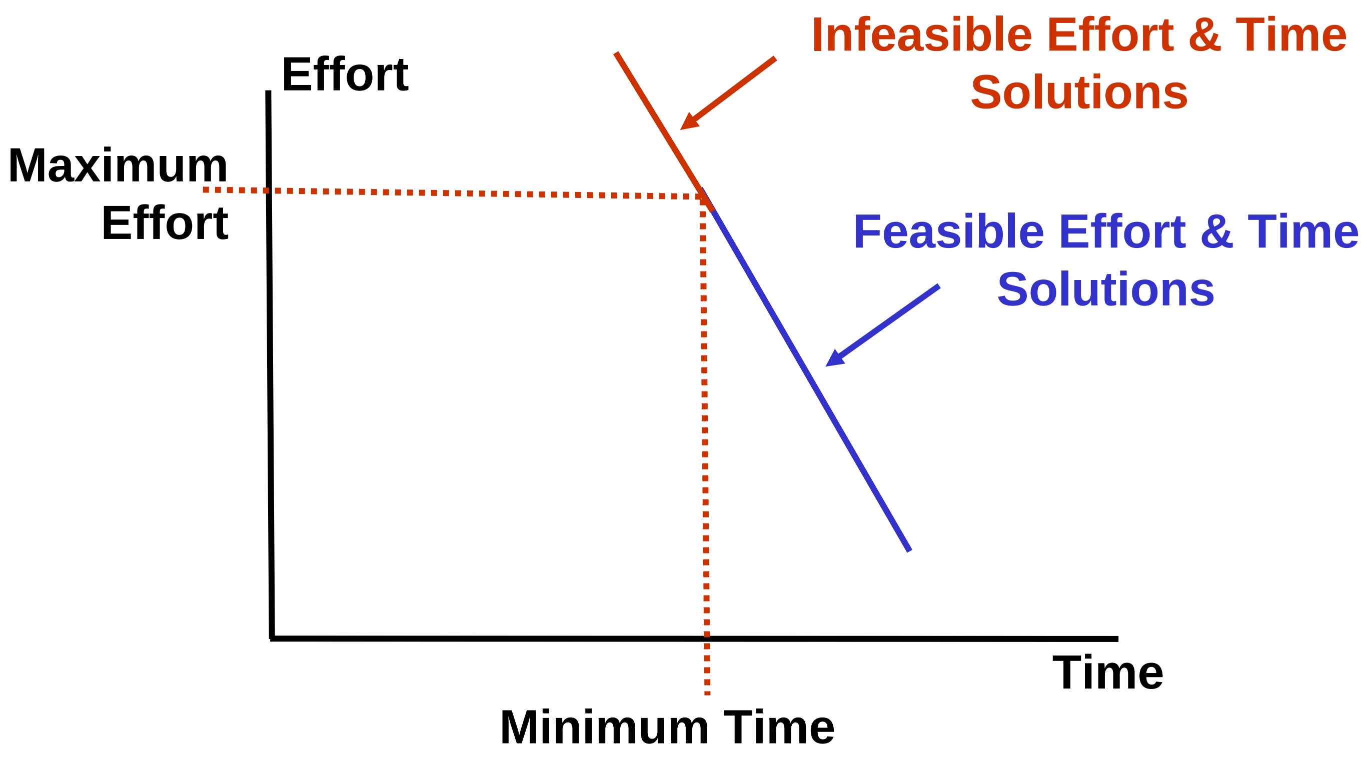
Two equations in two unknowns (E and T)

- E: effort in staff-months
- T: schedule in months

- $E \sim MBI * T^{3}$ (based on the **Norden-Rayleigh equation**)
- $E \sim (Size/PI)^{3} * T^{(-4)}$ (based on **Putnam's software equation**)
- The input parameters of the equations are:
  - Size: estimated product size (expressed in source lines of code, function points, or other size units; and
  - MBI: a Manpower Buildup Index that reflects the estimated rate of staff build-up for the project;
  - PI: the Productivity Index. Local data can be used to calibrate the PI parameter or industry-average values for different types of products can be used
- The output:
  - combinations of effort and time for given Size, MBI, and PI

adjustment factors are also provided

Simultaneous Solution of the SLIM Equations


Infeasible Effort & Time Solutions
Effort
Maximum Effort
Feasible Effort & Time Solutions
Time
Minimum Time

Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Software Size
- Pragmatic Estimation Techniques
- Parametric Estimation Models
  - Theory-Based Estimation Models
  - Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

A Typical Regression Equation

Effort = $a*(Size)^b$ * CPLX

Some examples:

Effort = $3.2\ (50)^{1.25} * 1.0 = 425$ staff-months

Effort = $3.2\ (50)^{1.25} * 0.8 = 340$ staff-months

Effort = $3.2\ (50)^{1.25} * 1.2 = 510$ staff-months

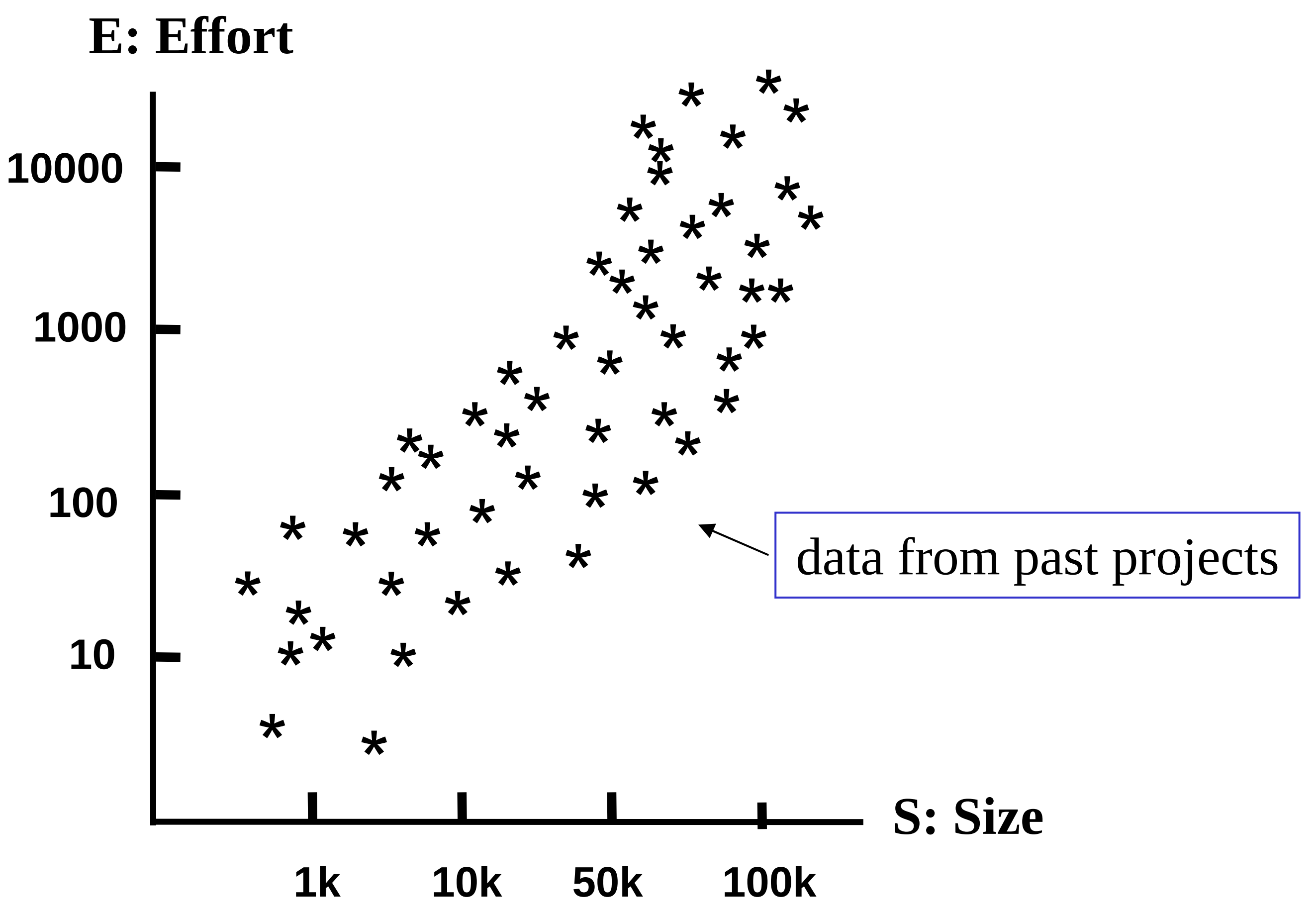
Notes:

1. Size is the future product attribute
2. Historical data is summarized in the exponential equation: $a*(Size)^b$
3. Problem and solution complexity (CPLX) is an adjustment factor

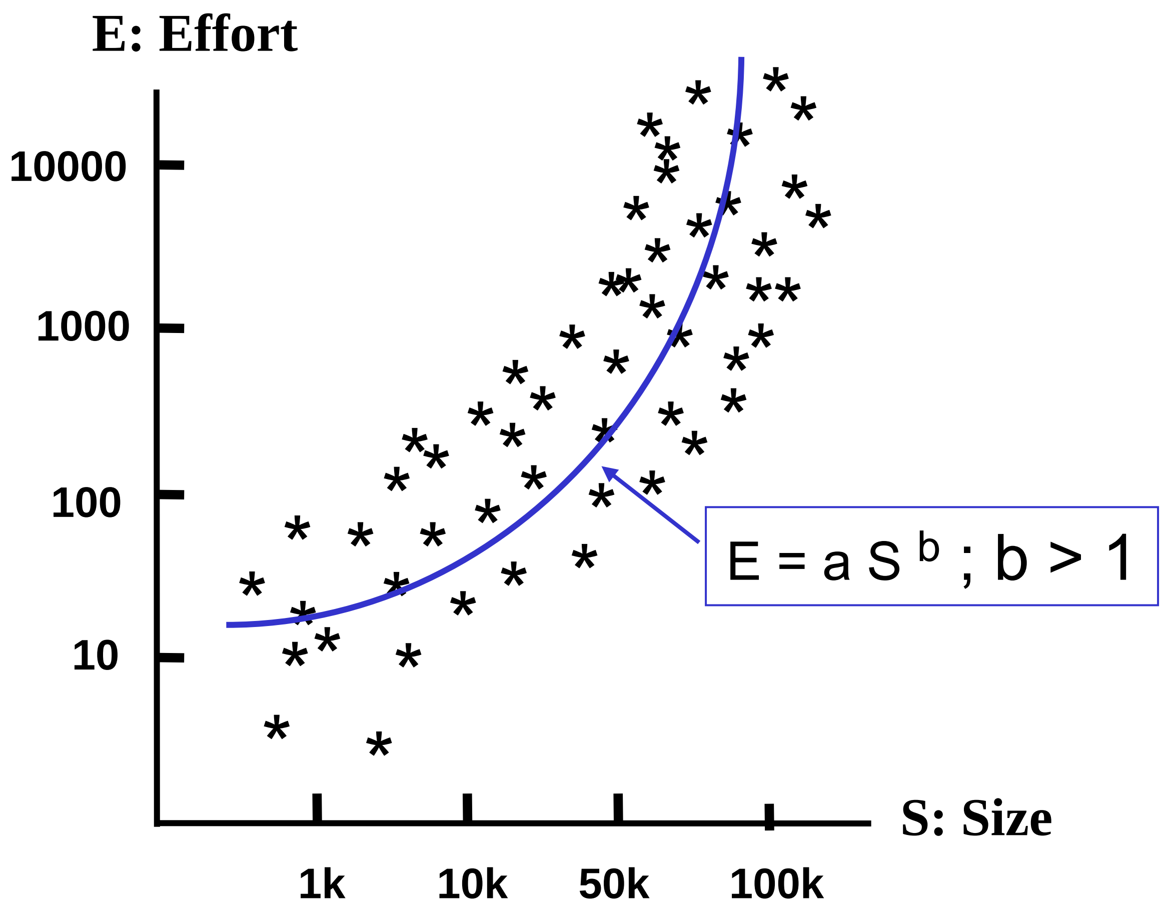
Some Typical Adjustment Factors for Similar Projects

- Complexity of the problem and the solution
- Requirements volatility
- Required reliability
- Development environment
- Target environment
- Methods and tools
- Personnel ability and experience
- Application experience
- Schedule constraint

A Scatter Plot of Effort versus Size for Past Projects



A Regression Equation Based on a Scatter Plot



$E = a S^b ; b > 1$

Q: what is the significance of b > 1?

Non-linear Scaling of Effort and Schedule Equations

$\text{Effort} = A * (\text{Size})^{B}$

Rule of Thumb: $B \sim 0.9 – 1.2$

What factors would explain $B > 1$?

What factors would explain $B < 1$?

$\text{Schedule} = C * (\text{Effort})^{D}$

Rule of Thumb: $D \sim 0.33 – 0.5$

$2.5*(50)0.33 \approx 9.1$

$2.5(100)0.33 \approx 11.4$

What is the significance of $D < 1$?

What factors would explain $D < 1$?

Adjustment Factors for Regression-based Estimation Models

"Why do two products of the same size differ by an order of magnitude in effort required to develop them?"

- If Size were a perfect predictor of Effort every historical data point would lie on the line of the estimation equation and the Residual Error from the scatter plot would be zero
- Adjustment factors are used to compensate for the differences between "average" values and predicted values
  - o"average values" are those computed by the regression equation
    - which is the "average" fit to the historical data

COCOMO: The Best-Known Regression Models*

- COCOMO: COnstructive COst MOdel
- COCOMO81: developed by Dr. Barry Boehm while at TRW around 1980
  - Documented in *Software Engineering Economics* by Barry Boehm, Prentice Hall, 1981
- Users' Group founded in 1985; R. Fairley, chairman until 1993
- Updated in 1987 for incremental development and the Ada process model
- COCOMO II was released in 1998 & updated in 2000
  - Documented in *Software Cost Estimation with COCOMO II* by Barry Boehm, et al, Prentice Hall, 2000

* COCOMO is a family of estimation models

Some Terminology

- In the COCOMO models the adjustment factors are called cost drivers
- The values assigned to the cost drivers are called effort multipliers

An Effort Equation with Effort Multipliers (EMs)

$\text{Effort} = 3.3 * (\text{Size})^{1.25} * EM_1 * EM_2 \ldots * EM_n$

or

$\text{Effort} = 3.3 * (\text{Size})^{1.25} * EAF$

Where EAF = Π (EMs)

And Π (EMs) is the product of Effort Multipliers

Suppose EM = CPLX * RELY * SCED

Where CPLX = 0.9; RELY = 1.4; SCED = 1.2

Then AF = 0.9 * 1.3 * 1.1 = 1.3

And $\text{Effort} = 3.3 * (\text{Size})^{1.25} * 1.3$

where CPLX: product complexity; RELY: required reliability; SCED: schedule constraint

Some COCOMO81 Effort and Schedule Equations

$SM = 3.6 * (KDSI)^{1.20} * EAF$

$TDEV = 2.5 * (SM)^{0.32}$

where SM is effort in Staff Months

KDSI is thousands of Delivered Source Instructions

EAF is the Effort Adjustment Factor

TDEV is the development schedule

COCOMO81 estimates Waterfall development
ADA COCOMO (1987) estimates incremental development
COCOMO II can be used either way

Fifteen COCOMO81 Cost Drivers

Cost Drivers		Ratings					
		Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes							
RELY:	Required Software Reliability	0.75	0.88	1.00	1.15	1.40	
DATA:	Data Base Size		0.94	1.00	1.08	1.16	
CPLX:	Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer Attributes							
TIME:	Execution Time Constraint			1.00	1.11	1.30	1.65
STOR:	Main Storage Constraint			1.00	1.06	1.21	1.56
VIRT:	Virtual Machine Volatility*		0.87	1.00	1.15	1.30	
TURN:	Computer Turnaround Time		0.87	1.00	1.07	1.15	
Personnel Attributes							
ACAP:	Analyst Capability	1.46	1.19	1.00	0.86	0.71	
AEXP:	Applications Experience	1.29	1.13	1.00	0.91	0.82	
PCAP:	Programmer Capability	1.42	1.17	1.00	0.86	0.70	
VEXP:	Virtual Machine Experience*	1.21	1.10	1.00	0.90		
LEXP:	Programming Language Experience	1.14	1.07	1.00	0.95		
Project Attributes							
MODP:	Use of Modern Programming Practices	1.24	1.10	1.00	0.91	0.82	
TOOL:	Use of Software Tools	1.24	1.10	1.00	0.91	0.83	
SCED:	Required Development Schedule	1.23	1.08	1.00	1.04	1.10	

*For a given software product. The underlying virtual machine is the complex of hardware and software (OS, DBMS, etc.). It calls on to accomplish its tasks.

A COCOMO81 EXAMPLE

Estimated size = 50,000 KDSI

Cost Driver	Situation	Rating	Effort Multiplier
RELY	Local Use of System, No Serious Recovery Problems	Nominal	1.00
DATA	20,000 Bytes	Low	0.94
CPLX	Communications Processing	Very High	1.30
TIME	Will Use 70% of Available Time	High	1.11
STOR	45K of 64K Store (70%)	High	1.06
VIRT	Based on Commercial Microprocessor Hardware	Nominal	1.00
TURN	Two-Hour Average Turnaround Time	Nominal	1.00
ACAP	Good Senior Analysts	High	0.88
AEXP	Three Years	Nominal	1.00
PCAP	Good Senior Programmers	High	0.86
VEXP	Six Months	Low	1.10
LEXP	Twelve Months	Nominal	1.00
MODP	Most Techniques in Use Over One Year	High	0.91
TOOL	At Basic Minicomputer Tool Level	Low	1.10
SCED	Nine Months	Nominal	1.00
	Effort Adjustment Factor (Product of Effort Multipliers)		1.17

AN EXAMPLE (continued)

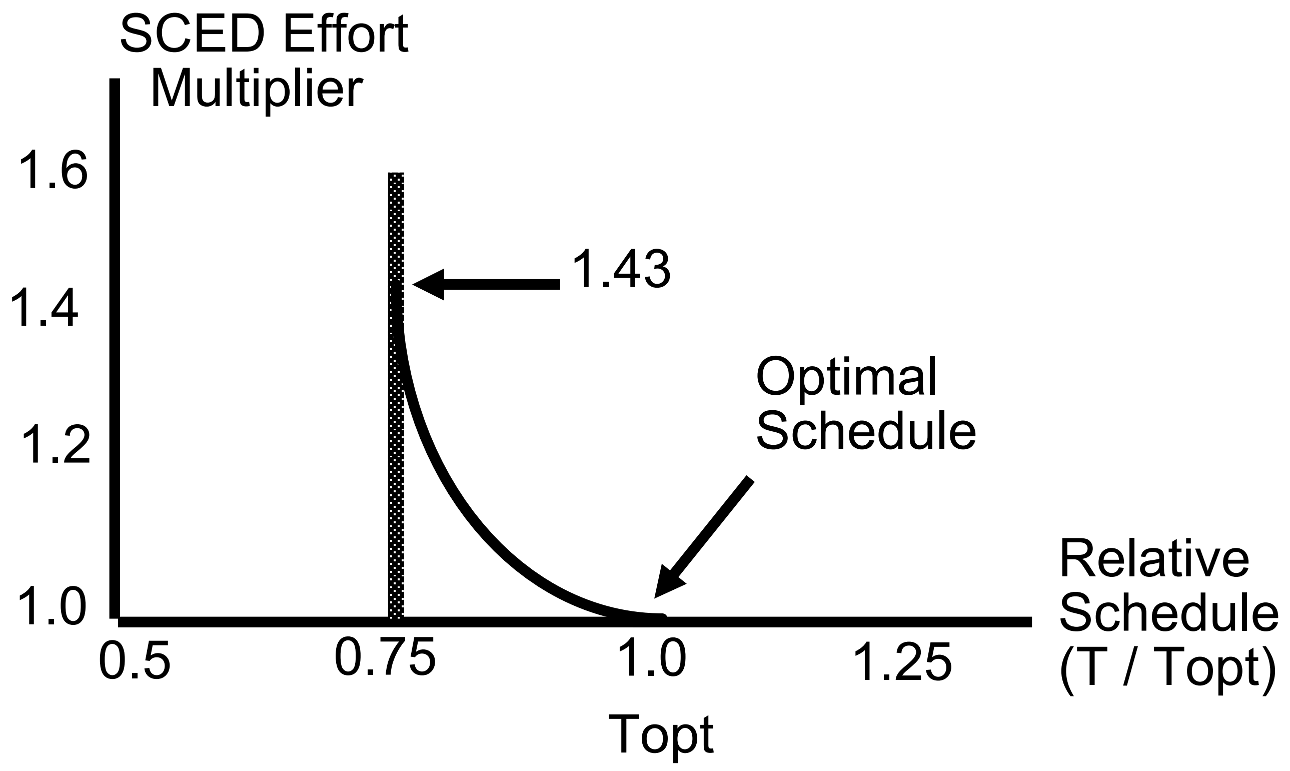
- Effort = $2.8 * (50)^{1.20} = 306.1$ SM
- EAF = 1.17
- Adjusted Effort = 306.1 * 1.17 = 358.2 SM
- Schedule = $2.5 * (358.2)^{0.32} = 16.4$ MO
- Average Staff Level = 358.2 / 16.4 ~ 22 people
- Productivity = 50,000 / 358.2 ~ 140 LOC/SM
- Cost = $10,000 per SM * 358.2 SM = $ 3.58 M

The SCED Effort Multiplier

- The effort multiplier for SCED was recalibrated from 1.23 in COCOMO81 to 1.43 in COCOMO II
  - for compressing the schedule by 25%
- And with no penalty for extending the schedule in COCOMO II

The total number of effort multipliers was extended from 15 to 17 in COCOMO II

The Effort Multiplier for Schedule Compression in COCOMO II



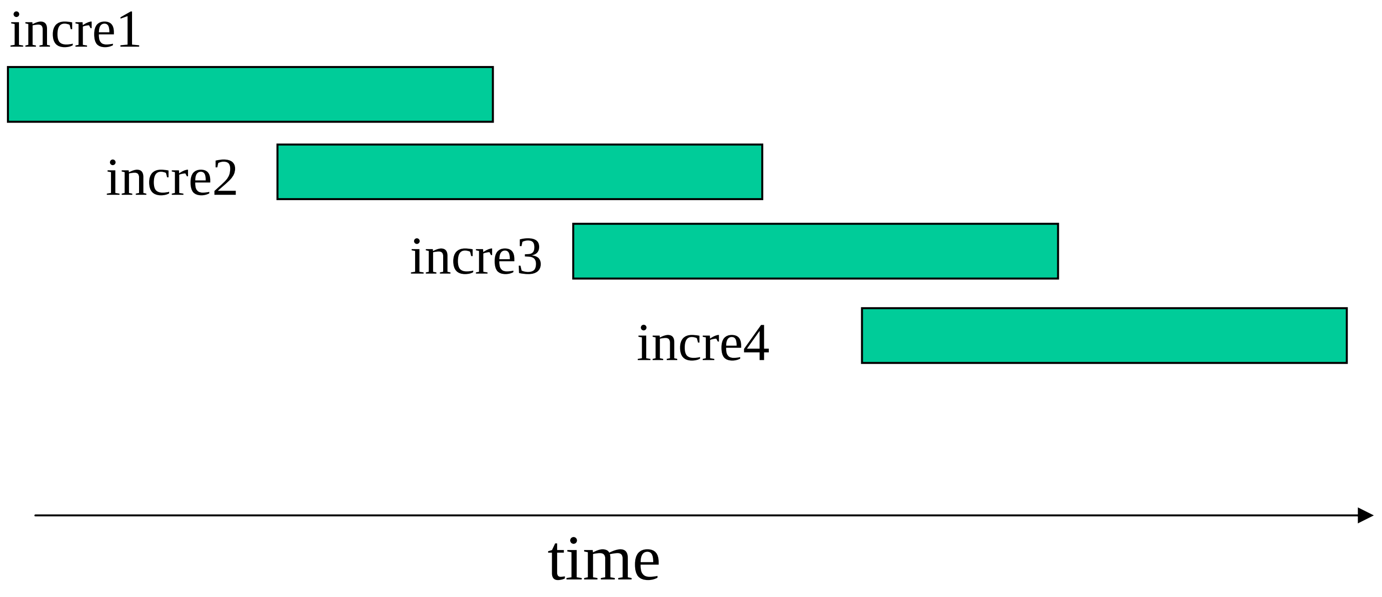
Topt is the optimal schedule computed using the COCOMO II Schedule equation

Adjusting the Number of Personnel for Schedule Compression in COCOMO II

- Suppose effort and schedule estimated for a project is 10 people for 12 months
  - estimated effort = 10 x 12 = 120 staff-months
- Now suppose the schedule is compressed by 25% to 9 months
  - T/Topt = 0.75
  - increased effort = 120 x 1.43 = 172 staff-months
  - required # personnel = 172 / 9 = 19 personnel

Compressing the schedule from 12 months to 9 months (by 25%) requires a (nearly) doubling of software developers

Estimation for Incremental Development (Ada COCOMO and COCOMO II)



- Estimation inputs are:
  - estimated size of each increment
  - adjustment factors for each increment
  - relative starting time of each increment
  - estimated breakage of each increment

The COCOMO II Effort Estimation Equation

- Effort in Person-Months (PM) is estimated using an equation of the form:

  $$PM = 2.94 * (SIZE)^{B} * \Pi\,(17 \text{ Effort Multipliers})$$

- The exponent B is of the form

  $$B = 0.91 + 0.01 \times \Sigma\,(\text{five Scale Factors})$$

- The exponent scale factors are:

PREC:	precedentedness of the system
FLEX:	flexibility in the requirements
RESL:	degree of risk resolution and interface specification at design review
TEAM:	cohesiveness of the development team
PMAT:	process maturity rating

The COCOMO II Effort Exponent and Constant

- The exponent B is of the form

  $B = 0.91 + 0.01 \times \Sigma$ (5 Scale Factors)

- The scale factors sum up to a value in the range of 0 to ~ 31.6

  B is thus in the range of 0.91 to 1.23

- The value of A, the effort equation constant, based on calibration on 83 projects, is 2.45
- **$PM = 2.94 \times [SIZE]^{B} \times \Pi$ (17 Effort Multipliers)**
- B ranges from 0.91 to 1.23

COCOMO II Cost Drivers (1)

Product Factors

- RELY: required reliability
- DATA: ratio of data size to code size
- CPLX: product complexity
- **RUSE: effort required for reuse***
- **DOCU: documentation match to lifecycle needs***

Platform Factors

- TIME: execution time constraint on target processor
- STOR: memory constraint on target processor
- PVOL: platform volatility

*** New in COCOMO II**

COCOMO II COST DRIVERS (2)

Personnel Factors

- ACAP: analyst capability
- PCAP: programmer capability
- APEX: application experience
- PLEX: platform experience
- LTEX: language and tools experience
- **PCON: personnel continuity***

Project Factors

- TOOL: use of software tools
- **SITE: multiple development sites***
- SCED: required development schedule

*** New in COCOMO II**

Some Other Attributes of COCOMO II

- COCOMO provides estimates of
  - total effort and schedule
  - effort and schedule by development phase
  - effort and schedule for 7 kinds of work activities within each development phase
  - monthly milestones, costs, and costs to date
  - early estimates (pre-architecture)
  - refined estimates (post-architecture)
- COCOMO II supports estimates for
  - incremental development
  - reuse of software components
  - developing components for reuse
  - function points or lines of code

COCOMO II Web Sites

- Information about COCOMO II, the annual Estimation Conference, and other COCOMO topics can be obtained at:

  sunset.usc.edu/research/COCOMOII/index.html

- A demo version of CoStar 7.0 can be downloaded from

  www.softstarsystems.com

  Costar includes a calibration tool and a COCOMO II estimation tool

  A production version of CoStar can be purchased from Softstarsystems

A CAUTION

- COCOMO II is calibrated to “industry averages” for many companies doing many different types of software development
    - **these averages may or may not be appropriate for your project and your company**
- Accurate estimates result from collecting project data within your organization and calibrating the model to your situation

Probabilistic Estimates

- A PERT approach can be used with parametric models such as SLIM and COCOMO to produce probabilistic estimates

PERT Estimation

- Three estimates are given for product size and for each of the cost drivers
  - o a: smallest likely value (e.g. at 20% probable value*)
  - o m: most probable value (e.g., 50% probable value*)
  - o b: largest likely value (e.g., 80% probable value*)

  calculate the expected value, E, and the standard deviation, σ of size and of each cost driver:

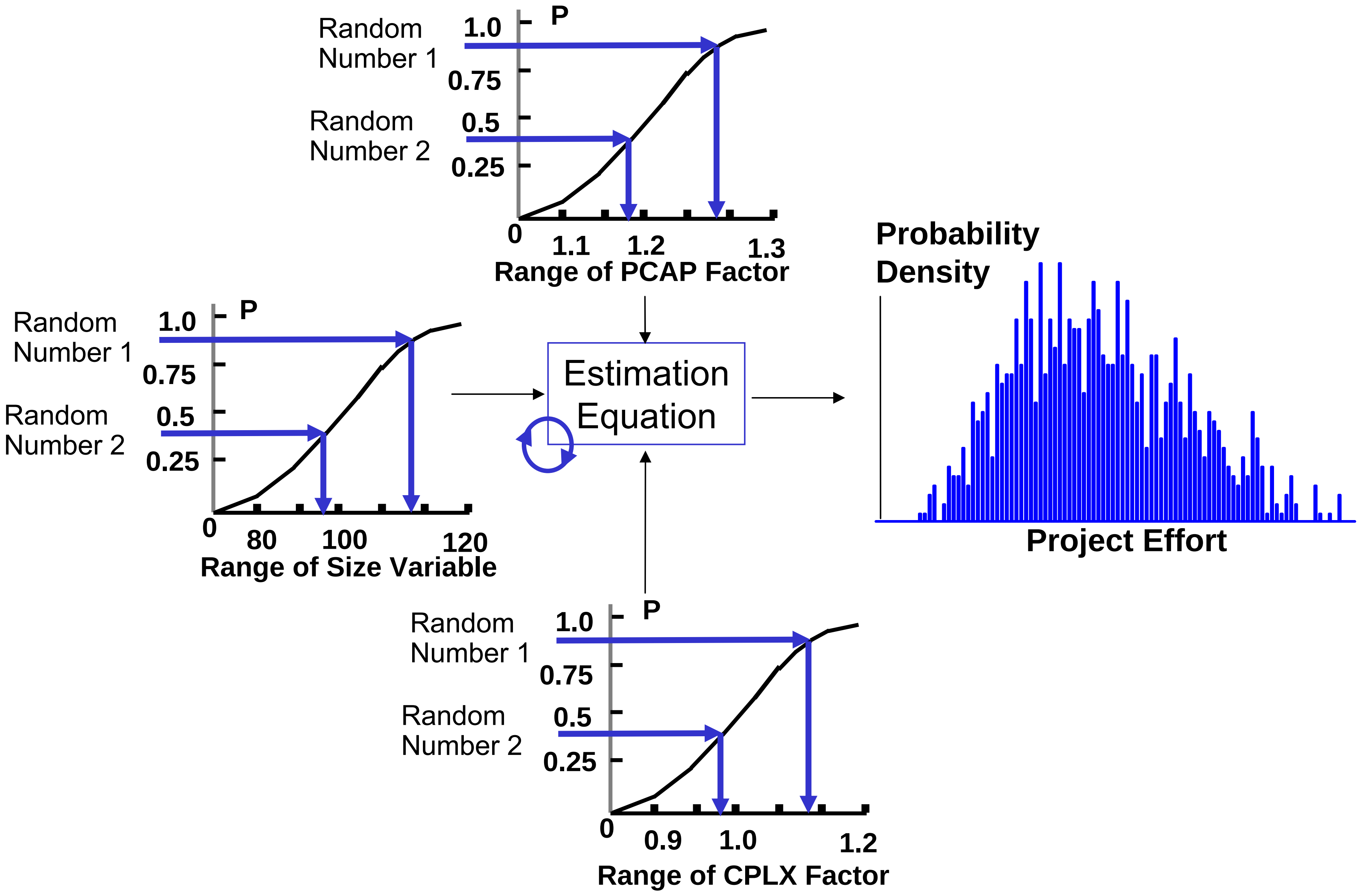
  $E = (a+4m+b)/6$ & $\sigma = (b - a)/3.2$
- Use Monte Carlo simulation to estimate the probability distribution of estimated project effort

* value not bigger than

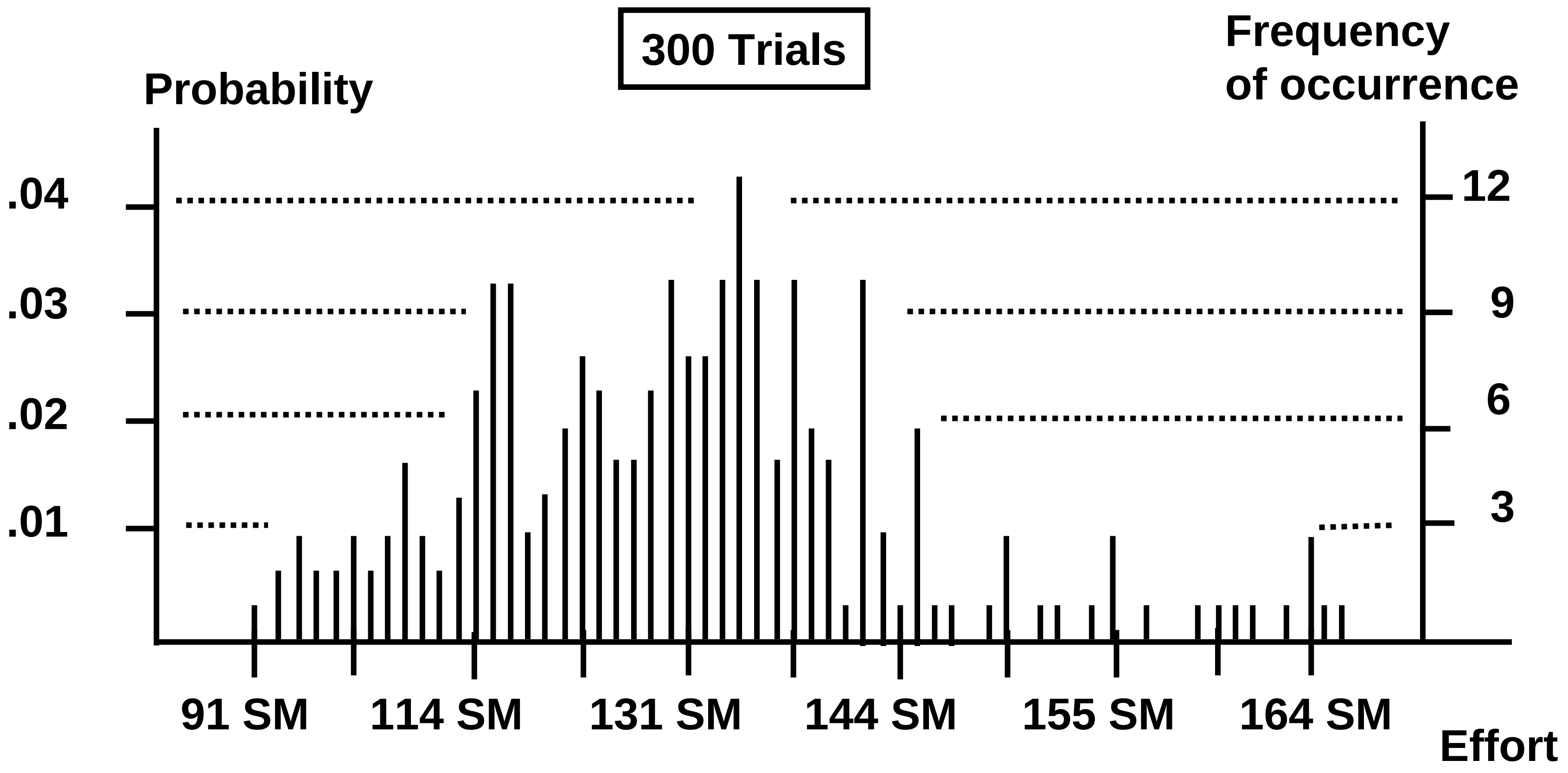
Monte Carlo Estimation

- Random numbers are used to select one value from each of the probability distributions
- Those numbers are used to compute one estimate
- The process is repeated a few hundred or a few thousand times
- The result is a probability density function

An Illustration of Monte Carlo Estimation



A Probabilistic Effort Estimate (1)



A Probabilistic Effort Estimate (2)

- On the previous slide, the number of values to the left of any value of effort ratio-ed to the total number of values is the probability the project can be completed with that amount of effort or less
- For example, approximate 270 of 300 trials on the previous slide are to the left of 144 staff-months
  - thus, it is estimated that it is 90% probable the project can be completed with 144 staff-months of effort
  - the square root ROT would provide an estimate of 12 people for 12 months at 90% probability

Chapter 6 Topics

- Fundamental Principles of Estimation
- Designing to Project Constraints
- Estimating Software Size
- Pragmatic Estimation Techniques
- Parametric Estimation Models
  - Theory-Based Estimation Models
  - Regression-Based Estimation Models
- Estimation Tools
- Estimating Life Cycle Resources, Effort, and Cost
- An Estimation Procedure
- A Template for Recording Estimates

An Estimation Procedure

An estimation procedure is specified in Section 6.11 of the textbook

Documentation of an Estimate Should Include the Following

1. Project identifier
2. Version number and date of the estimate
3. Total estimate effort
4. Total estimated schedule
5. Name(s) of the estimator(s)
6. Rationale for the estimate (i.e., why is this estimate being made? feasibility, initial estimate, periodic update, aperiodic update, etc)
7. Elements changed (for updates to an estimate)
8. Amount of time and effort spent in making the estimate
9. Estimation methods and tools used
10. The basis of estimation for each method or tool used (industry averages, expert judgment, local historical data, etc)
11. A list of assumptions made for each method or tool used
12. A list of constraints observed in making the estimates

continued on next slide

Documentation of an Estimate Should Include the Following (2)

13. A list of inputs used for each method or tool used

    (e.g., size, PI, MBI, for SLIM; size and adjustment factors for COCOMO)
14. Other estimation data provided by the estimation method or tool

    (e.g., project milestones, effort for various project activities by project phase, estimated pre-release and post-release defects, estimated reliability at product delivery, total lifecycle costs)
15. A range of estimates for effort, schedule, resources, cost, and quality attributes with associated probabilities for each method or tool used
16. Risk factors for the project
17. The estimator's level of confidence in the accuracy of the estimate

    (0 to 10; low, medium, high)
18. Information, resources, and time needed to make an improved estimate

The Main Points of Chapter 6 (1)

- A project estimate is a projection from past to future, suitably adjusted to account for differences between past and future.
- All estimates are based on a set of assumptions that must be realized and a set of constraints that must be satisfied.
- Projects must be re-estimated periodically as understanding grows and aperiodically as project parameters change.
- Size is the primary variable in most software estimation models.
- Traditional size measures include lines of code and function points.
- External size measures (ESMs) can be developed for each application area.

The Main Points of Chapter 6 (2)

- Estimation models can be categorized as pragmatic, theory-based, and regression-based.
- Theory-based and regression-based estimation models can be calibrated using local data.
- Software estimation tools provide a variety of capabilities.
- Estimates should be prepared using at least two different methods.
- Estimates should be documented using a standard template.
- CMMI, ISO, IEEE, and PMI provide frameworks, standards, and guidelines for project estimation techniques (see Appendix 6A to Chapter 6).