SESSION 2 OUTLINE

- Pragmatic Estimation: Checkpoint
- Estimating Product Attributes: SSM
- Theory-based Estimation: SLIM
- Regression-based Estimation: COCOMO
- An Estimation Procedure
ALGORITHMIC TECHNIQUES

- Algorithmic techniques can be categorized as:
  - pragmatic: based on rules of thumb and industry trends
  - theory-based: based on mathematical relationships among project factors
  - regression-based: based on equations derived from historical data

- Algorithmic techniques incorporate:
  - product characteristics: size, complexity, reliability, etc.
  - process characteristics: personnel, methods, tools, etc.

CHECKPOINT - A PRAGMATIC ESTIMATION TOOL

- Checkpoint is thought to be a pragmatic estimation tool; we assume the internal algorithms are based on industry averages and trends
- Checkpoint runs on DOS and UNIX platforms
- Checkpoint is marketed by SPR, Inc.:
  SPR, Inc.
  77 S. Bedford St.
  Burlington, MA 01803-5154
  (508) 273-0140
CHECKPOINT FEATURES

- Three sizing modes:
  - function points
  - feature points
  - lines of code
- Two estimation modes
  - quick estimate: 10 input screens
  - detailed estimate: 30+ screens

QUICK ESTIMATE INPUTS

- Project description: name, estimator
- Project nature: new, enhancement, repair
- Project scope: prototype, standalone, ...
- Project class: personal use ... major release
- Project type: application ... real time
- Estimation goal: standard ... highest quality
- Programming language
- Complexity: problem, code, data
- Size: function points, feature points, LOC
- Unit costs: labor rates, etc.
SOME CHECKPOINT OUTPUTS
(12 screens of output)

- Estimate summary:
  - inputs
  - development schedule
  - effort (person-months)
  - number of staff
  - pages of documentation
  - cost
- Quality: delivered defects
- Reliability: CPU hours to failure
- Productivity: KLOC/SM

CHECKPOINT RISK FACTORS

- Detailed input to Checkpoint allows setting of 15 risk factors
- Each factor is rated on a scale of 0 to 5
  - 0: factor not applicable
  - 5: very high risk
- The Checkpoint risk factors are listed on the next chart
CHECKPOINT RISK FACTORS

- risk of high project novelty
- risk of unstable user requirements
- risk of change in product architecture
- risk of change in development hardware
- risk of inadequate speed or memory capacity
- risk of inadequate functionality
- risk of poor quality and reliability
- risk of significant usability problems
- risk of significant schedule overrun
- risk of significant cost overrun
- risk of insufficient project staffing
- risk of insufficient project skill levels
- risk of excessive schedule pressure
- risk of high staff turnover and attrition
- risk of major management disagreements

Session 2 Outline

- Pragmatic Estimation: Checkpoint
  ➔ Estimating Product Attributes: SSM
- Theory-based Estimation: SLIM
- Regression-based Estimation: COCOMO
- An Estimation Procedure

In general, we are not much better at estimating product attributes than we are at estimating effort and schedule
WHY ESTIMATE PRODUCT ATTRIBUTES?
(Why Not Estimate Effort and Schedule Directly?)

Some reasons:
• standardized measures of product attributes permit comparisons among projects and organizations
• basing estimates on product attributes such as size, complexity, timing constraints, etc. provides a basis for making trade-offs among schedule, budget, and product attributes

WHY ESTIMATE PRODUCT ATTRIBUTES?
(continued)
• Product attributes provide inputs to cost estimation models and tools
  – e.g., COCOMO
    - product attributes
    - computer attributes
    - personnel attributes
    - project attributes
• Direct estimates of effort and schedule may be harder to justify than estimates based on product characteristics
WHAT PRODUCT ATTRIBUTES INFLUENCE EFFORT AND SCHEDULE?

- Size of code and data
- Type of system
- Complexity of the algorithms
- Performance requirements
- Reliability and safety requirements
- Host and target environments
- Design constraints
  - memory and timing
  - Interface requirements: software, hardware, people
  - languages and tools

WBS/PERT Sizing

Project X

Project Mgmt.  System Design  Software Engnrng  Hardware Engnrng  System Integr.

Analysis  Design  Coding  Test

Input  Edit  Process  Display
### WBS / PERT SIZING - II

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>m</th>
<th>b</th>
<th>E</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>INPUT</td>
<td>6k</td>
<td>10k</td>
<td>20k</td>
<td>11.0k</td>
<td>2.3 kloc</td>
</tr>
<tr>
<td>EDIT</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>7.5</td>
<td>1.5</td>
</tr>
<tr>
<td>PROCESS</td>
<td>8</td>
<td>12</td>
<td>19</td>
<td>12.5</td>
<td>1.8</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>8.0</td>
<td>1.3</td>
</tr>
</tbody>
</table>

\[ E = 39k \text{ and } sE = 3.5k \]
\[ E - sE = 35.5 \text{ and } E + sE = 42.5k \]

---

### WBS / PERT SIZING - III

- One Standard Deviation: 68% probable
  \[ E - sE = 35.5 \text{ kloc and } E + sE = 42.5 \text{ kloc} \]
- Two Standard Deviations: 97.5% probable
  \[ E - 2sE = 32 \text{ kloc and } E + 2sE = 46 \text{ kloc} \]
- Three Standard Deviations: 99% probable
  \[ E - 3sE = 28.5 \text{ kloc and } E + 3sE = 49.5k \]
SSM - THE SOFTWARE SIZING MODEL

- SSM is a statistical sizing model developed by George Bozoki
- SSM is marketed by:

  Target Software
  552 Shorebird Circle
  Suite 1202
  Redwood City, CA 94065
  (415) 941-8832

THE SOFTWARE SIZING MODEL (SSM)

- INPUTS:
  - Four rankings of modules sizes*
    - Rank order: (z, y, x)
    - Pairwise: (x, y); (z, x); (y, z)
    - Intervals: 50 < x, y < 100
      100 < z < 200
    - PERT: a, m, b for each module
  - At least two calibration modules
  - size can be in lines of code or function points
SSM OUTPUT

- OUTPUTS:
- Expected Size & Standard Deviation For Each Module
- System Size & Confidence Limits

<table>
<thead>
<tr>
<th>MODULE NAME</th>
<th>EXPECTED SIZE</th>
<th>STD DEV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telemetry</td>
<td>12900</td>
<td>2600</td>
</tr>
<tr>
<td>Operations</td>
<td>103700</td>
<td>17300</td>
</tr>
<tr>
<td>Simulator</td>
<td>8500</td>
<td>0000</td>
</tr>
<tr>
<td>Monitor</td>
<td>29900</td>
<td>6100</td>
</tr>
<tr>
<td>Status</td>
<td>13400</td>
<td>2000</td>
</tr>
<tr>
<td>Trends</td>
<td>12000</td>
<td>0000</td>
</tr>
</tbody>
</table>
ESTIMATING THE IMPACT OF OTHER PRODUCT ATTRIBUTES

Subsequently, we will see how to specify the impact of other product attributes, such as:

- Complexity of the algorithms
- Performance requirements
- Reliability and safety requirements
- Host and target environments
- Design constraints
  - memory and timing
  - interface requirements: software, hardware, people
HOW DO WE USE STATISTICAL ESTIMATES OF PRODUCT SIZE AND OTHER PRODUCT ATTRIBUTES?

- Select points on the probability distribution of size, for example, to determine effort and schedule for a given probable size.
- Monte Carlo simulation generalizes this idea by randomly selecting independent points from input probability distributions to produce histograms of output variables.

INVERSE SAMPLING OF A PROBABILITY DISTRIBUTION

Cumulative Probability

Size Estimate (KLOC)

Copyright © 1996 by SEMA, Inc.
### AN EXAMPLE OF MONTE CARLO OUTPUT

<table>
<thead>
<tr>
<th>Probability</th>
<th>300 Trials</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>.01</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>.02</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>.03</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>.04</td>
<td></td>
<td>12</td>
</tr>
</tbody>
</table>

382K 477K 572K 667K 762K 857K Dollars

80% probable @ $700K

---

### SLIM: A THEORY-BASED MODEL

- SLIM: The Software Lifecycle Model
- SLIM was developed by Larry Putnam in the 1970s and has been upgraded several times
- SLIM is marketed by QSM, Inc.
  
QSM, Inc.
1057 Waverley Way
McLean, VA 22101
(703) 790-0055


---

page 2-13
SLIM: The Software Lifecycle Model

SLIM is based on
1. the Norden-Rayleigh equation to model the staffing profile for a software project and
2. the Putnam software equation to relate product size, project effort, and project schedule to the NR equation


THE NORDEN - RAYLEIGH MODEL OF PROJECT STAFFING

\[ y' = 2Ke^{-at^2} \]
where \( a = \frac{1}{2t_d^2} \)

K: area

Copyright © 1996 by SEMA, Inc.
PUTNAM'S SOFTWARE EQUATION

Effort = (SLOC^3 / PI) * Time^{-4}

- Effort is in staff-months
- SLOC is software size in Source Lines of Code
- PI is a "productivity index"
  - PI is calibrated using local history, or set by SLIM using the estimation input parameters
- Time is in months

THE MANPOWER BUILDPUP INDEX (MBI)

MBI is a measure of the slope of the "rising edge" of the Norden-Rayleigh curve:

MBI can be determined from historical data or from a SLIM table

Copyright © 1996 by SEMA, Inc.

Estimation chart 2-29
SLIM EFFORT / SCHEDULE COMBINATIONS

The software equation and the MBI equation provide two equations in two unknowns (Effort and Time):

\[ \text{Effort} = B \times \left( \frac{\text{SLOC}^3}{\text{PI}} \right) \times \text{Time}^{-4} \]
\[ \text{Effort} = \text{MBI} \times \text{Time}^3 \]

or:

\[ \log \text{Effort} \propto -4 \times \log \text{Time} \]
\[ \log \text{Effort} \propto 3 \times \log \text{Time} \]
SOLUTION CONSTRAINED BY
Emax, Emin and Tmax, Tmin

\[ \log E = 3 \log t \]

\[ \log E = 4 \log t \]

feasible solutions:

SLIM ESTIMATION INPUTS

- Estimation inputs include:
  - PI and MBI
  - Size: can be broken down by module
  - Cost elements: salary rates, computer
  - Development environment
  - System constraints
  - Modern programming practices
  - Experience
  - Constraints on Time and Effort
SIMULATION / PROBABILITY (continued)

- Size is specified by the three Beta probability numbers (a,m,b)
- PI and MBI can be specified as known or uncertain
  - if known, single point values are used
  - if uncertain, probability distributions are used
- Solutions are computed using Monte Carlo simulation

AN EXAMPLE

ESLOC:  100K ± 30K  
PI:  13 ± 2  
MBI:  2 ± 15%

ESLOC: Estimated Source Lines of Code
AN EXAMPLE (continued)

Effort & Schedule Distribution

<table>
<thead>
<tr>
<th></th>
<th>50% probable</th>
<th>84% probable</th>
<th>97.5% probable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort</td>
<td>297 SM</td>
<td>443 SM</td>
<td>589 SM</td>
</tr>
<tr>
<td>Schedule</td>
<td>19.1 MO</td>
<td>22.3 MO</td>
<td>25.5 MO</td>
</tr>
</tbody>
</table>

Copyright © 1996 by SEMA, Inc.

Estimation chart 2-37
SLIM RISK ANALYSIS

<table>
<thead>
<tr>
<th>50% probable</th>
<th>84% probable</th>
<th>97.5% probable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort:</td>
<td>Schedule:</td>
<td></td>
</tr>
<tr>
<td>297 SM</td>
<td>19.1 MO</td>
<td></td>
</tr>
<tr>
<td>443 SM</td>
<td>22.3 MO</td>
<td></td>
</tr>
<tr>
<td>589 SM</td>
<td>25.5 MO</td>
<td></td>
</tr>
</tbody>
</table>

Effort Risk: In a 26 month schedule, it is 87% probable that the project can be completed with 500 staff-months of effort.

Schedule Risk: With 600 SM of effort, it is 93% probable that the project can be completed in 24 months.

ESTIMATING SOFTWARE RELIABILITY

- SLIM also provides estimates of software reliability.
- Reliability is reported as Mean Time To Defect (MTTD) at various milestones.
- MTTD is the reciprocal of the rate at which defects are discovered.
  e.g., 2 defects per day = 0.5 days MTTD.
ESTIMATED MTTD

- MTTD is estimated by a Rayleigh equation:
  \[ Em = \left(6 \frac{Er}{t_d^2}\right) t e^{-\left(3 t \frac{2}{2} t_d\right)} \]
  where:
  \( Er \) = total errors in life of the project
  \( Em \) = errors per month
  \( t \) = time in months
- \( MTTD = \frac{1}{Em} \)

* Measures for Excellence, page 128
  by L. Putnam and W. Myers, Prentice-Hall, 1992

Session 2 Outline

- Pragmatic Estimation: Checkpoint
- Estimating Product Attributes: SSM
- Theory-based Estimation: SLIM
  \( \Rightarrow \) Regression-based Estimation: COCOMO
- An Estimation Procedure
COCOMO: THE BEST-KNOWN REGRESSION MODEL

- COCOMO: CONstructive COst MOdel
- Developed by Dr. Barry Boehm while at TRW
- Users' Group founded in 1985
- Updated in 1987 for incremental development and the Ada process model
- COCOMO 2.0 now being developed

REGRESSION-BASED COST MODELING

$log E_{(Effort)}$ vs $log S_{(Size)}$

$E = a \cdot S^b$

slope $b$

Copyright © 1996 by SEMA, Inc.

Estimation chart 2-44
LINEAR REGRESSION
IN THE LOGARITHMIC DOMAIN

- \[ \log \text{Effort} = \log a + b \cdot \log \text{Size} \]
  \[
  \log \text{Effort} = 0.5 + 1.25 \cdot \log \text{Size}
  \]
  \[
  \text{Effort} = 3.3 \cdot (\text{Size})^{1.25}
  \]

- \[ \log \text{Schedule} = \log c + d \cdot \log \text{Effort} \]
  \[
  \log \text{Schedule} = 0.4 + 0.33 \cdot \log \text{Effort}
  \]
  \[
  \text{Schedule} = 2.8 \cdot (\text{Effort})^{0.33}
  \]

COST DRIVERS FOR
REGRESSION-BASED MODELS

"what causes two projects of the same size to differ by an order of magnitude in required effort?"

- If Size were a perfect predictor of Effort every data point would lie on the line of the regression equation and the Residual Error would be zero
- Cost Drivers are used to compensate for the differences between predicted and actual values
COCOMO EFFORT ADJUSTMENT FACTOR

- The Effort Adjustment Factor (EAF) is the product of 15 effort multipliers
- The estimate produced by the effort equation is multiplied by EAF to produce an adjusted estimate
- Adjusted effort is used to compute the adjusted schedule using the regression equation for TDEV

THE FIFTEEN COST DRIVERS

- Product Characteristics
  - RELY required software reliability
  - DATA data base size
  - CPLX product complexity
- Computer Attributes
  - TIME execution time constraint
  - STOR main storage constraint
  - VIRT virtual machine volatility *
  - TURN computer turnaround time

*the virtual machine is the hardware and software (OS, compiler, DBMS, etc) used to develop the software
FIFTEEN COST DRIVERS (continued)

- Personnel Attributes
  - ACAP analyst capability
  - AEXP applications experience
  - PCAP programmer capability
  - VEXP virtual machine experience
  - LEXP programming language experience

- Project Attributes
  - MODP modern programming practices
  - TOOL use of software tools
  - SCED required development schedule

---

A COCOMO EXAMPLE

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Situation</th>
<th>Rating</th>
<th>Effort Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>Local Use of System, No Serious Recovery Problems</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>DATA</td>
<td>32,000 Bytes</td>
<td>Low</td>
<td>0.94</td>
</tr>
<tr>
<td>CPLX</td>
<td>Communications Processing</td>
<td>Very High</td>
<td>1.20</td>
</tr>
<tr>
<td>TIME</td>
<td>Will Use 70% of Available Time</td>
<td>High</td>
<td>1.11</td>
</tr>
<tr>
<td>STOR</td>
<td>64K of 64K Store (70%)</td>
<td>High</td>
<td>1.85</td>
</tr>
<tr>
<td>VRT</td>
<td>Based on Commercial Microprocessor Hardware</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>TURN</td>
<td>Two-Hour Average Turnaround Time</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>ACAP</td>
<td>Good Senior Analysts</td>
<td>High</td>
<td>0.80</td>
</tr>
<tr>
<td>AEXP</td>
<td>Three Years</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>PCAP</td>
<td>Good Senior Programmer</td>
<td>High</td>
<td>6.88</td>
</tr>
<tr>
<td>VEXP</td>
<td>Six Months</td>
<td>Low</td>
<td>1.10</td>
</tr>
<tr>
<td>LEXP</td>
<td>Twelve Months</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
<tr>
<td>MODP</td>
<td>Most Techniques in Use Over One Year</td>
<td>High</td>
<td>0.91</td>
</tr>
<tr>
<td>TOOL</td>
<td>At Basic Minicomputer Tool Level</td>
<td>Low</td>
<td>1.10</td>
</tr>
<tr>
<td>SCED</td>
<td>Nine Months</td>
<td>Nominal</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Effort Adjustment Factor (Product of Effort Multipliers) 1.17
A COCOMO EXAMPLE (continued)

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

THE CB-COCOMO TOOL

- CB-COCOMO is a Monte Carlo implementation of COCOMO
- Crystal Ball and CB-COCOMO are marketed by:
  Decisioneering, Inc.
  1380 Lawrence Street, Suite 520
  Denver, CO 80204 - 9849
- Crystal Ball and CB-COCOMO run on DOS and Macintosh computers
MONTE CARLO SIMULATION USING CRYSTAL BALL

- Crystal Ball is a general purpose Monte Carlo simulation tool
- Crystal Ball runs in conjunction with a spreadsheet
- Simulation models are constructed as spreadsheet equations
- Probability distributions, rather than single values, are specified as the input values in the spreadsheet cells
- Random sampling is used to produce an output histogram

CB-COCOMO

- CB-COCOMO is a Monte Carlo spreadsheet implementation of COCOMO
- Those size and cost drivers for which you are uncertain can be specified by input probability distributions
- Size and cost drivers for which you are more certain can be input as point values
CB-COCOMO OUTPUT

Probability  300 Trials  Frequency

<table>
<thead>
<tr>
<th>Probability</th>
<th>382K</th>
<th>477K</th>
<th>572K</th>
<th>667K</th>
<th>762K</th>
<th>857K</th>
</tr>
</thead>
<tbody>
<tr>
<td>.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>.03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Dollars

80% probable @ $700K

CUMULATIVE PROBABILITY OF COST

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>50th</td>
<td>$570k</td>
</tr>
<tr>
<td>75th</td>
<td>$620k</td>
</tr>
<tr>
<td>85th</td>
<td>$667k</td>
</tr>
<tr>
<td>95th</td>
<td>$762k</td>
</tr>
</tbody>
</table>

It is 84% probable, for example, that the project can be completed for $665K or less, under the stated assumptions and conditions.
Session 2 Outline

- Pragmatic Estimation: Checkpoint
- Estimating Product Attributes: SSM
- Theory-based Estimation: SLIM
- Regression-based Estimation: COCOMO

⇒ An Estimation Procedure

AN ESTIMATION PROCEDURE

1. Determine the purpose of and required accuracy for the estimate
2. Identify the required information and sources for it
3. Plan the schedule, resources, and responsibilities for developing the estimate
4. Validate the software requirements
5. Develop a first-cut software architecture
AN ESTIMATION PROCESS - II

6. Develop a work breakdown structure
7. Estimate the size, scope, complexity, and desired quality attributes for each component
8. Supply any additional factors required by the estimation techniques to be used (always use more than one estimation technique, including a WBS-based approach)
9. Prepare estimates using the selected estimation techniques

AN ESTIMATION PROCESS - III

10. Conduct sensitivity analyses on the estimates
11. Reconcile differences in the estimates
12. Identify risk factors and risk mitigation strategies
13. Prepare a final estimate, including assumptions, probable ranges, confidence level, and contingency plans
AN ESTIMATION PROCESS - IV

14. Prepare a plan for updating the estimate at periodic intervals and on aperiodic events
15. Collect project data and retain a history file
16. Re-estimate cost and schedule in accordance with step 14

A STANDARD FORMAT FOR ESTIMATES

Estimates should contain:
- Name(s) of the estimator(s)
- A clearly stated basis of estimation
- Estimated effort, schedule, resources
- A list of cost drivers and assumptions
  - (both positive and negative)
- A range of estimates with associated probability levels
- Risk factors for the project
- The estimator’s level of confidence in the accuracy of the estimate
  - (0 to 10; low, medium, high)
A RISK MANAGEMENT PARADIGM

SESSION 2 SUMMARY

- Pragmatic Estimation: Checkpoint
- Estimating Product Attributes: SSM
- Theory-based Estimation: SLIM
- Regression-based Estimation: COCOMO
- An Estimation Procedure