

# Using Monte Carlo and COCOMO-2 to Model a Large IT System Development

COCOMO/SCM 17

October 24, 2002

John Bailey

Institute for Defense Analyses

# Study Goals

- Estimate the cost and schedule of a large IT system assuming Forté, COBOL, and Java development approaches
  - Model the probability distribution of results
  - Give 50% and 80% estimates
- Use recent experience to calibrate estimates
  - CSCI-1 and CSCI-2 are finished
  - Seeking estimates for CSCI-3 and CSCI-4

# Estimation Methodology

- CSCI-3 and CSCI-4 cost estimation methods depend on development approach
  - 1) Forté
    - Extrapolate from recent experience developing related software modules
  - 2) COBOL
    - Use COCOMO
  - 3) Java or other new languages
    - Make do with limited experience and reports from early adopters

# Outline

- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) Description of probabilistic estimation with examples
- 3) Distribution of CSCI-4 cost estimates
- 4) COCOMO modeling of CSCI-4 cost and schedule if using COBOL
- 5) Calibrating productivity of Java to COBOL
- 6) Summary table of results

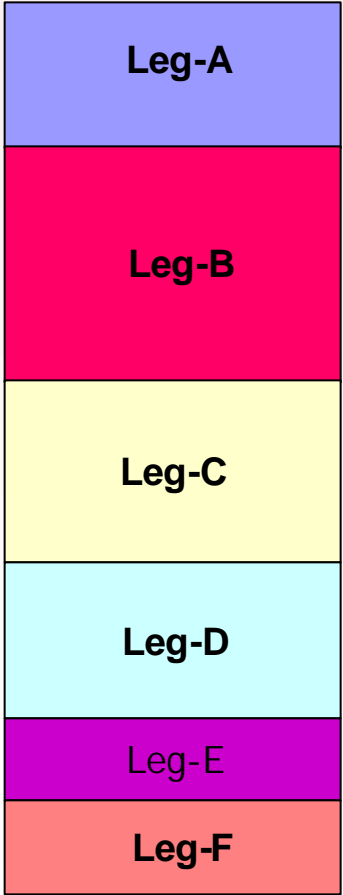
# Forté Productivity Now Known

Module	Status	Forté Size	Cost \$M
CSCI-1	Complete	615,494	27.3 \$44/LOC
CSCI-2	Complete	338,792	15.2 \$45/LOC
CSCI-3	Half Complete	?	?
CSCI-4	Just Beginning	?	?

We would need Forté size estimates for CSCI-3 and CSCI-4

# Constituency of CSCI-4 Legacy Lines

Legacies Doing CSCI-4 Functions	Legacy Lines of Code (LOC)	Percentage Attributed to CSCI-4	Legacy LOC Doing CSCI-4 Functions
Leg-A	1,080,663	83%	901,354
Leg-B	1,602,713	90%	1,448,444
Leg-C	1,155,235	97%	1,119,689
Leg-D	1,628,774	59%	967,410
Leg-E	500,000	100%	500,000
Leg-F	1,900,000	31%	582,667
Total	7,867,385	70%	<b>5,519,564</b>



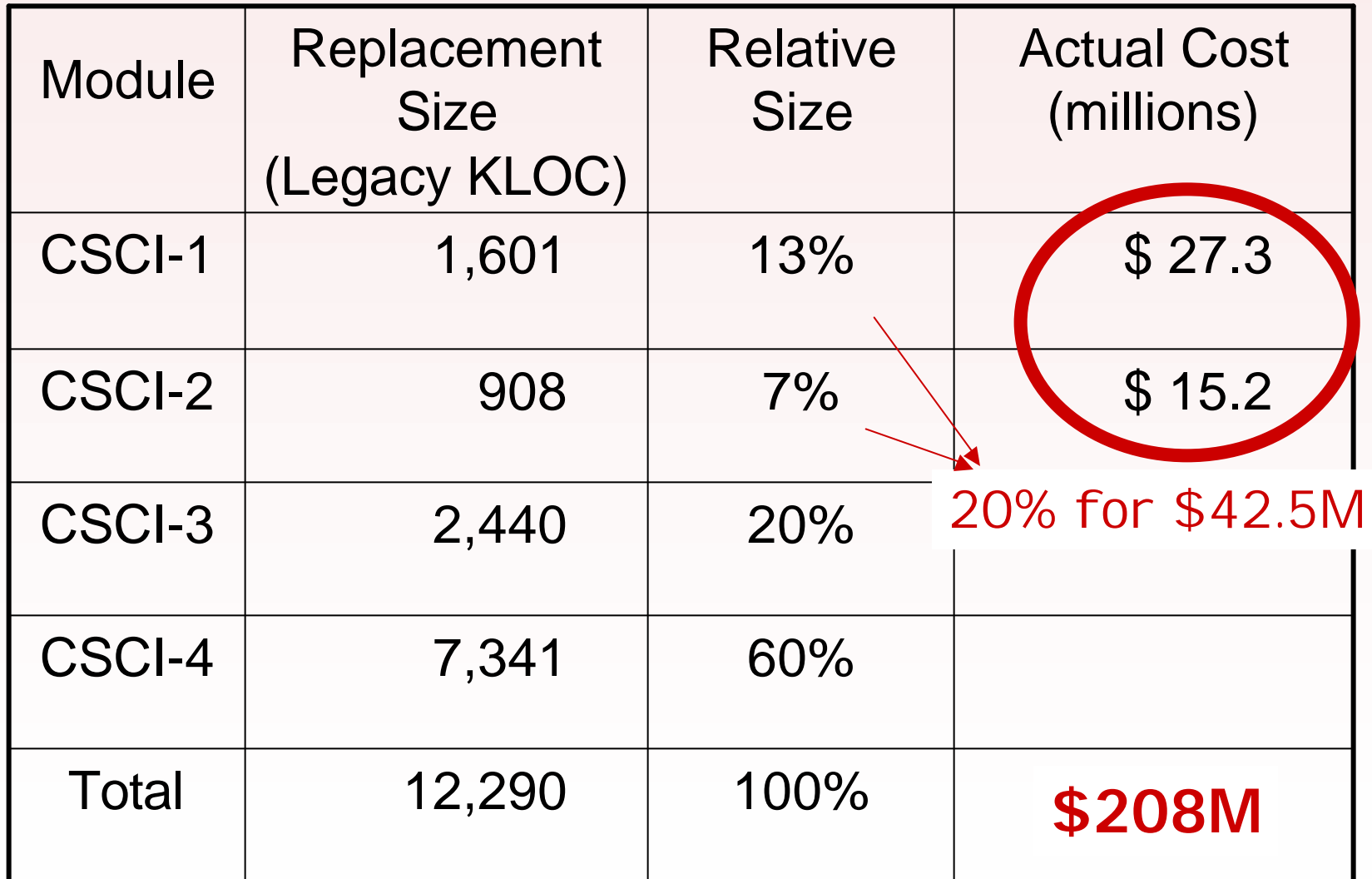
# Counts of Legacy Lines Replaced

Module	Legacy Lines (KLOC)	Additional Functions	Replacement Size (Legacy KLOC)
CSCI-1	1,280	25%	1,601
CSCI-2	160	470%	908
CSCI-3	1,525	60%	2,440
CSCI-4	5,520	33%	7,341
Total	8,485		12,290

We can measure size and productivity with “legacy lines replaced”

# Extrapolating Cost of Replacement

Module	Replacement Size (Legacy KLOC)	Relative Size	Actual Cost (millions)
CSCI-1	1,601	13%	\$ 27.3
CSCI-2	908	7%	\$ 15.2
CSCI-3	2,440	20%	20% for \$42.5M
CSCI-4	7,341	60%	
Total	12,290	100%	<b>\$208M</b>



# Estimating CSCI-3 and CSCI-4 Costs

Module	Replacement Size (Legacy KLOC)	Relative Size	Actual Cost (millions)
CSCI-1	1,601	13%	\$ 27.3
CSCI-2	908	7%	\$ 15.2
CSCI-3	2,440	20%	<b>\$ 41M</b>
CSCI-4	7,341	60%	<b>\$124M</b>
Total	12,290	100%	<b>\$208M</b>

Module	Replacement Size (Legacy KLOC)	Relative Size	Actual Cost (millions)
CSCI-1	1,601	13%	\$ 27.3
CSCI-2	908	7%	\$ 15.2
CSCI-3	2,440	20%	\$ 41
CSCI-4	7,341	60%	\$ 124
Total	12,290	100%	\$ 208

# Outline

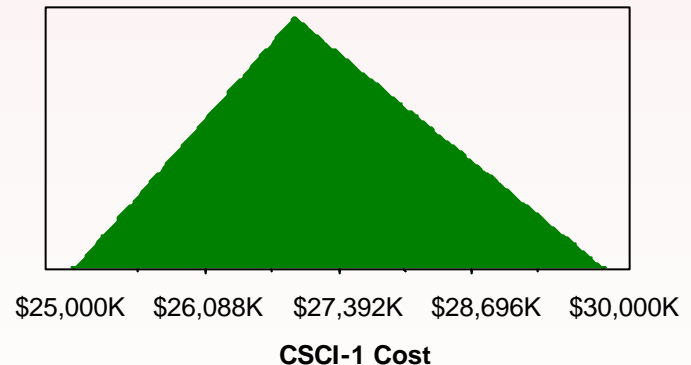
- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) **Description of probabilistic estimation with examples**
- 3) Distribution of CSCI-4 cost estimates
- 4) COCOMO modeling of CSCI-4 cost and schedule if using COBOL
- 5) Calibrating productivity of Java to COBOL
- 6) Summary table of results

# Input and Output Ranges

- Model inputs are, themselves, estimates
  - In previous point-estimate, we assumed:
    - ✓ Experience levels remain constant
    - ✓ No additional learning curve improvements
    - ✓ Legacy lines are a good proxy to compare CSCI-1/CSCI-2 effort with CSCI-3/CSCI-4 effort
    - ✓ Line counts and excess function estimates are correct
  - What if we assign a range of possible values to each input?
- Input uncertainty results in output uncertainty

# Combining Many Uncertain Inputs

- We usually have some idea about the likely range of values for each input
  - CSCI-1 cost between **\$25M** and **\$30M** but probably close to **\$27M**



- Can we compose distribution estimates for many inputs into an output distribution?
- Yes, either analytically or through simulation

# Monte Carlo Simulation

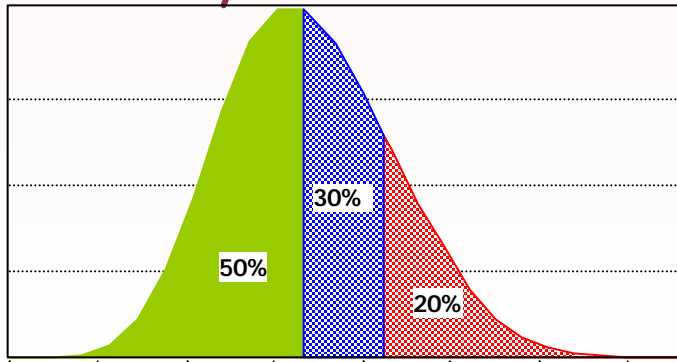
- In reality, every input has a unique (though unknown) value, leading to a single result
  - The exact input values can't be known early on
  - We want to try to predict the output when we only have a general idea about likely input distributions
  - Since some input combinations are more likely than others, some output values are more likely
- A Monte Carlo simulation computes a distribution of output values by picking sets of input values according to their probability
  - We are not modeling model inaccuracy, we are modeling input uncertainty

# Outline

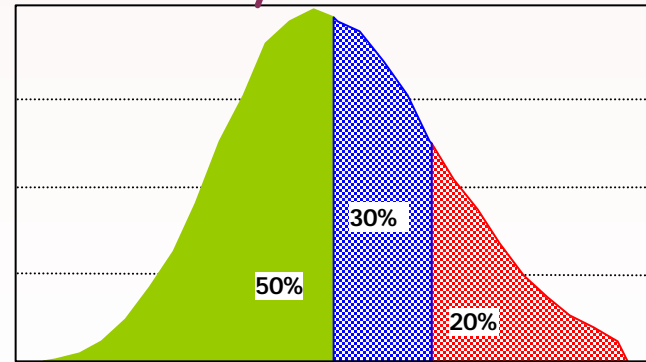
- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) Description of probabilistic estimation with examples
- 3) **Distribution of CSCI-4 cost estimates**
- 4) COCOMO modeling of CSCI-4 cost and schedule if using COBOL
- 5) Calibrating productivity of Java to COBOL
- 6) Summary table of results

# Modeling the Size of CSCI-1

Module	Legacy Lines (KLOC)	Additional Functions	Implemented Size
CSCI-1	1,280 <b>1,275 - 1,375</b>	25% <b>15% - 35%</b>	1,601 (point) <b>1,600 - 1,731</b>
CSCI-4	5,520	33%	7,341 (point)



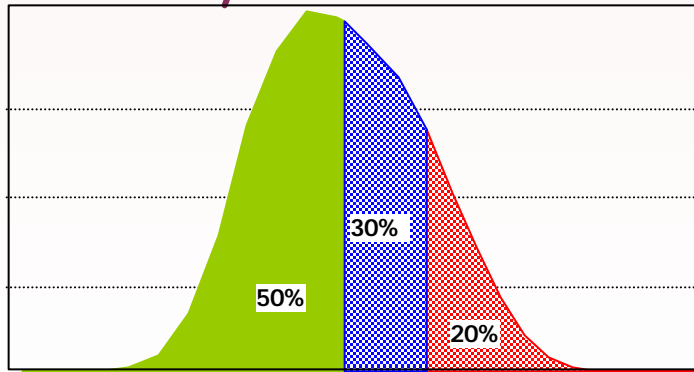
900,000 1,125,000 1,350,000 1,575,000  
CSCI-1 Legacy Lines Replaced



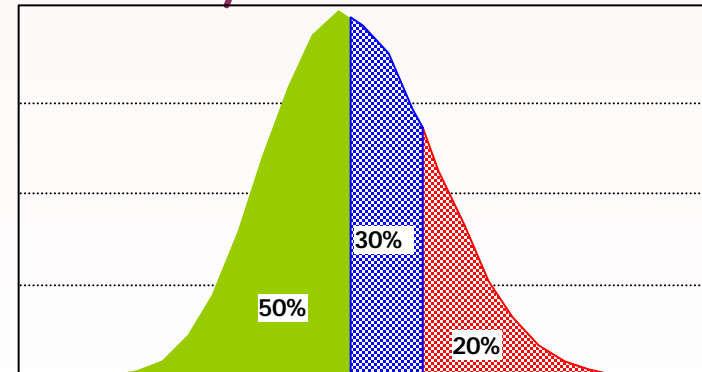
1,200,000 1,400,000 1,600,000 1,800,000 2,000,000  
CSCI-1 Size in Equivalent Legacy Lines of Code

# Modeling the Size of CSCI-4

Module	Legacy Lines (KLOC)	Additional Functions	Implemented Size
CSCI-1	1,280 1,275 - 1,375	25% 15% - 35%	1,601 1,600 - 1,731
CSCI-4	5,520 5,500 - 5,695	33% 23% - 43%	7,341 7,320 - 7,632



CSCI-4 Legacy Lines Replaced

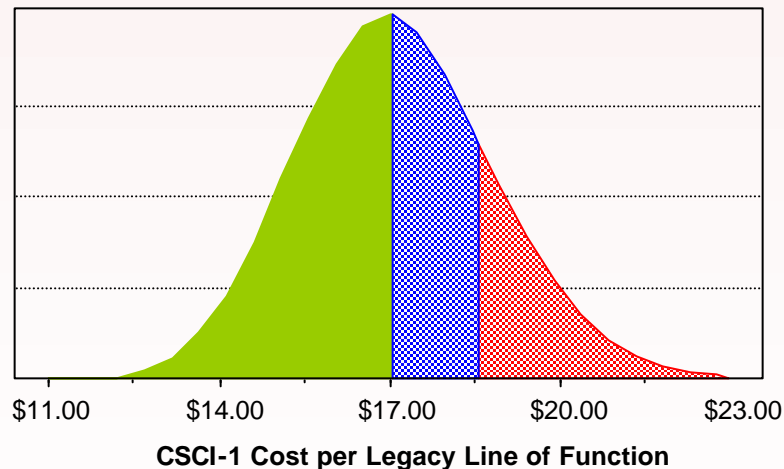


CSCI-4 Size in Equivalent Legacy Lines of Code

# Estimated Cost per Legacy Line

- CSCI-1 cost between \$25M and \$30M
- 50%: CSCI-1 implemented (as many as) 1,600 KLOC
- 80%: CSCI-1 implemented (only) 1,469 KLOC
- Composite result:

Cost / Size :



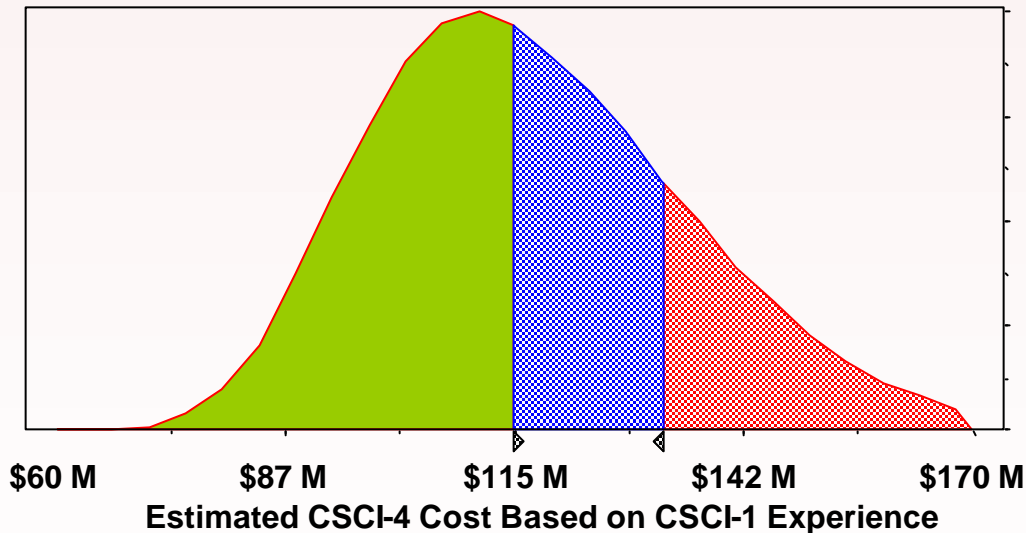
- One line of legacy code was implemented by CSCI-1 for \$17.04 (50%) or \$18.56 (80%)

# Allowing for Learning

- The final result includes an estimate of productivity improvement over time
  - If using the same tools, we would expect the team to know them better and avoid past mistakes
  - If using new tools, we would expect them to be superior due to advances in technology
  - Historically, a 4% annual improvement is typical
- Learning factor distribution
  - Assumed a range with an average cost that is 93% of the historical average (7% improvement)
    - Allowed from 25% worse to 30% better productivity

# Modeling the Cost of CSCI-4

- Compose cost-per-legacy-line estimates for CSCI-1 with legacy line size estimates of CSCI-4
- Compose with learning estimates



- CSCI-4 will cost < \$115M 50% of the time
- CSCI-4 will cost < \$133M 80% of the time

# Estimating a Schedule for CSCI-4

- Calibrate to CSCI-1 schedule experience
  - 2,725 PM expended over 41 months (D)
  - D=41 is 91% of COCOMO predicted schedule
  - We can scale this experience up to the 50<sup>th</sup> and 80<sup>th</sup> percentile estimates for CSCI-4 effort

Module	Simulation Percentile	Effort (PM)	COCOMO Duration (months)	Adjusted Duration Prediction
CSCI-1	--	2,725	45	41 actual
CSCI-4	50th	11,509	72	65
CSCI-4	80th	13,284	75	68

# Outline

- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) Description of probabilistic estimation with examples
- 3) Distribution of CSCI-4 cost estimates
- 4) **COCOMO modeling of CSCI-4 cost and schedule if using COBOL**
- 5) Calibrating productivity of Java to COBOL
- 6) Summary table of results

## Approach 2: CSCI-4 in COBOL

- Previous modeling assumed CSCI-4 would be developed in Forté
- If development occurs in COBOL, we can not base estimates on prior experience
- We used COCOMO II equations in a Monte Carlo simulation to obtain a ranges of estimates

# Estimating CSCI-4 with COCOMO II

- To predict effort from size using COCOMO II

$$\text{Staff\_Months} = 2.94 \cdot \text{EAF} \cdot \text{KLOC} \cdot \text{Process\_Adjustment\_Exp}$$

*EAF* factors included APEX (2), PLEX (2), LTEX (2), PCAP (2), TOOL (3), SCED (3)

*Process\_Adjustment\_Exp* factors included CMM (2, 1+, 1-) and precedentedness (somewhat, generally, largely)

- Modeled a distribution of values for each factor above  
Net factor was 83% of nominal cost (17% better than industry average)

# COBOL CSCI-4 Effort and Schedule

- COCOMO II simulation results assume:
  - Replacement of CSCI-4 legacies
  - Addition of about 1/3 more functionality
  - Estimated net COBOL size to be at least 3MLOC but more likely 5MLOC

Simulation Percentile	Labor Months	Duration (months)	Annual (millions)	Average Staff	Cost (millions)
50th	27,800	75	\$44	370	\$278
80th	37,300	84	\$53	444	\$372

# Excerpt from Crystal Ball-Driven Excel Worksheet

COCOMO-2 Runs for CSCI-4 using Distributions of Input Values (Monte Carlo simulations)

This version (CSCi4cocomosims8.xls) correctly uses the before-compression effort to estimate schedule.

App Exp.	0.92	1	0.88						
Platf. Exp.	0.955	1	0.91						
Lang Exp.	0.94	1	0.91			familiarity	Exponents	CMM level	
Prog. Capab.	0.94	1	0.88			neutral	1.0997	level two	
Tools	0.89	1	0.9	0.78		familiar	1.0873	level two	
Schedule	1.202500972	1	1.14	1.43		very familiar	1.0749	level two	
Duration Effect	0.828447941	100%	85%	75%		neutral	1.1153	level one high	
						familiar	1.1029	level one high	
						very familiar	1.0905	level one high	
	<i>Effort</i>	<i>Schedule</i>				neutral	1.1309	level one low	
Coefficient	2.94	3.67				familiar	1.1185	level one low	
Effort Adjust	0.830849473	*see notes below to create graphs					very familiar	1.1061	level one low
Exponent	1.1029	0.3179							
Labor Rate	\$120,000								
Hours / Month:	152								

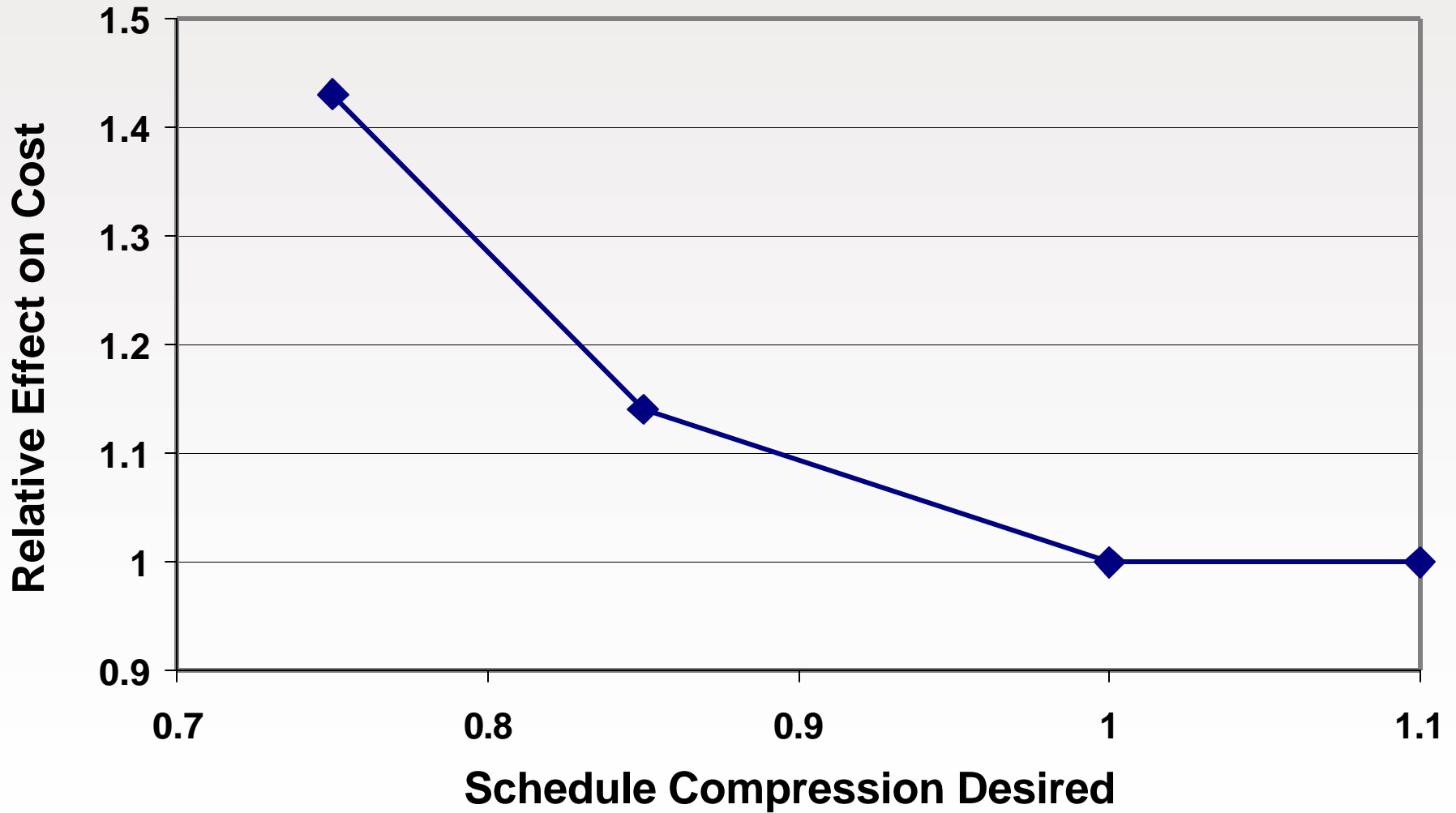
Monte Carlo workspace:

Size KSLOC	Labor Months	Labor Years	Duration (months)	Ave. Staff	Ave. Budget	Total Cost	Productivity LOC/hr
5,000	29,340	2,445	75.5	389	\$ 46,662,296	\$293 M	1.12
Confidence	Labor Months	Labor Years	Duration (months)	Ave. Staff	Ave. Budget	Total Cost	Productivity LOC/hr
50%	27,832	2,319	75.29	370	\$ 44,362,749	\$278 M	1.18
80%	37,250	3,104	83.86	444	\$ 53,302,359	\$372 M	0.88

# Effects of Schedule Compression

- Modeled results assumed 17% schedule compression to produce an acceptable completion date (0.83 x nominal duration)
- Premium for this compression was 20% (1.2 x nominal cost)
- COCOMO II allows up to a 25% compression for a 43% premium

# COCOMO II Compression Effects



# Outline

- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) Description of probabilistic estimation with examples
- 3) Distribution of CSCI-4 cost estimates
- 4) COCOMO modeling of CSCI-4 cost and schedule if using COBOL
- 5) **Calibrating productivity of Java to COBOL**
- 6) Summary table of results

# Approach 3: Build CSCI-4 with Java

- Cost, relative to COBOL, driven by two assumptions
  - How many Java lines are needed to compute the function of 5MLOC of COBOL?
  - How much do Java programmers cost relative to COBOL programmers?
- Researched industry experiences with Java
  - Found large variance: between 40% and 90% reduction in developed code relative to COBOL
    - Presumably caused by many first-users in data
  - Assumed the mid-point: 65% compression
  - Examined attributes of highly successful projects achieving up to 90% compression

# Results of Java Modeling

- Conservatively modeled a 46% savings relative to COBOL
- Removed schedule compression applied in COBOL model
  - Less incentive to pay a premium to compress time
  - Removed the 20% cost premium

Simulation Percentile	Labor Months	Duration (months)	Annual (millions)	Average Staff	Cost (millions)
50th	7,481	66	\$23	113	\$125
80th	10,022	73	\$28	138	\$168

# Variability with Java Approach

- Reports of Java experiences varied greatly
- Assumed average of reported productivity
  - Probably skewed by many early users
- By controlling factors that are associated with Java success, we might expect better results
  - Team experience, tool use, Java library assets
- Java is probably representative of other modern, high-reuse technologies
  - E.g., Result was very similar to Forté result

# Outline

- 1) Point-estimates of CSCI-3 and CSCI-4 costs using CSCI-1 and CSCI-2 experiences
- 2) Description of probabilistic estimation with examples
- 3) Distribution of CSCI-4 cost estimates
- 4) COCOMO modeling of CSCI-4 cost and schedule if using COBOL
- 5) Calibrating productivity of Java to COBOL
- 6) **Summary table of results**

# Summary of Results

Approach	Cost at 50 <sup>th</sup> Percentile (millions)	Cost at 80 <sup>th</sup> Percentile (millions)	Duration at 50 <sup>th</sup> Percentile	Duration at 80 <sup>th</sup> Percentile
Forté	\$ 115	\$ 133	65 months	68 months
COBOL	\$ 278	\$ 373	80 months	88 months
Java	\$ 125	\$ 168	66 months	73 months

# Monte Carlo Illustration: Simple Effort Estimation Example

- We expect a project's productivity to be **1.1 LOC/Hr**, although it could be as low as **0.5 LOC/Hr** or as high as **1.3 LOC/Hr** (triangle)
- We expect the size of the project to be as low as **70,000 LOC** or as high as **150,000 LOC** but don't know anything about likelihood (flat)
- The worst case would be  $150,000 \div 0.5 = 300,000$  hours, best case would be  $70,000 \div 1.3 = 54,000$  hours
- However, a Monte Carlo simulation tells us that **50%** of the time, the project will take **114,000 hours** (or less) and **80%** of the time, it will take **144,000 hours** (or less)

