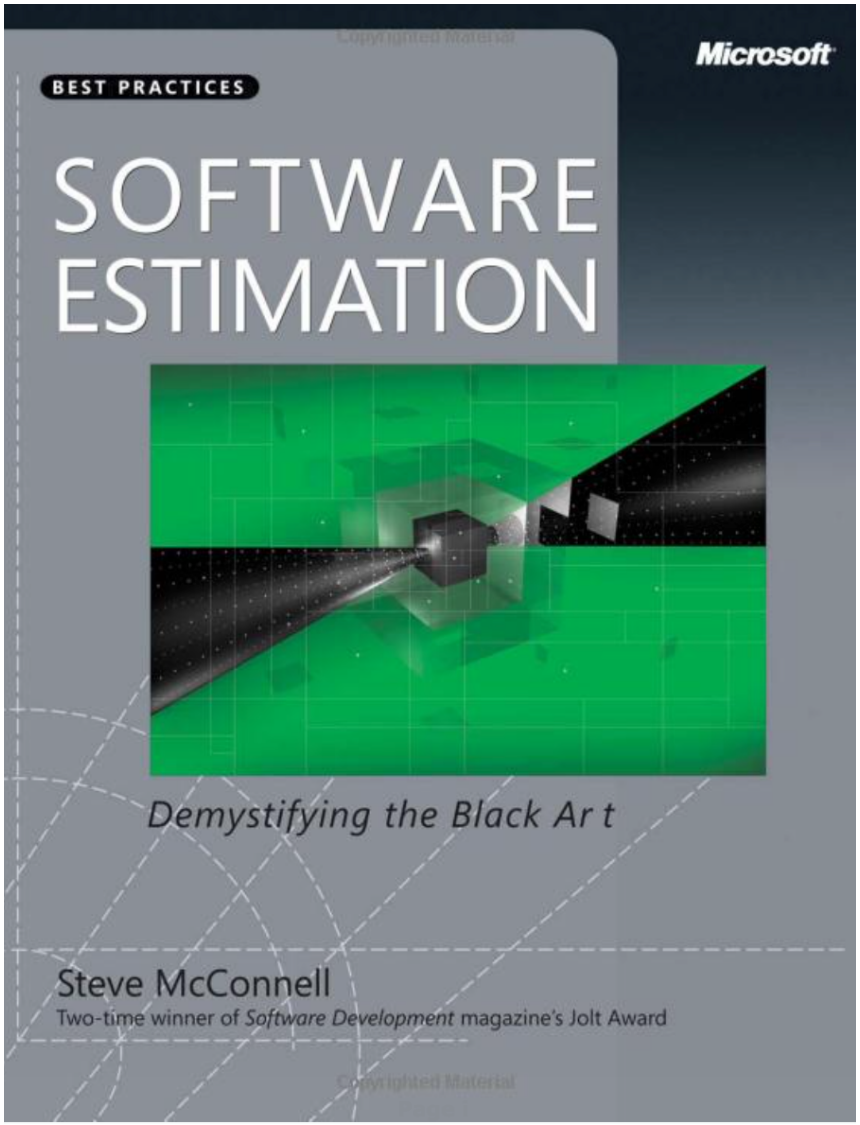
SE463
Software Requirements Specification & Analysis

Software Estimation

Readings:

McConnell, S., *Software Estimation: Demystifying the Black Art,* 2006.

Sources



Lecture content comes from

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Software Cost Estimation

Given an early description of the system, you want to determine as early as possible if a proposed system or requirement is technically and economically feasible. Economically feasible means whether the client is willing to pay what it will cost to develop the project, and whether the developer is willing to devote the resources to the project. Both of these questions have to be answered based on estimates, estimates of the cost of the project and estimates of the development effort.

Why Estimate Software Cost and Effort?

- To assess economic feasibility
- To understand resource needs
- To provide a basis for agreeing to a job
- To make commitments that we can meet

Terminology

An estimate is a prediction of how long a project will take or how much it will cost.

A target is a statement of a desirable business objective. It is generally a goal set at a performance level and then pursued.

A commitment is a promise to deliver.

"do not assume that the commitment has to be the same as the estimate; it doesn't."

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Inaccurate Estimates

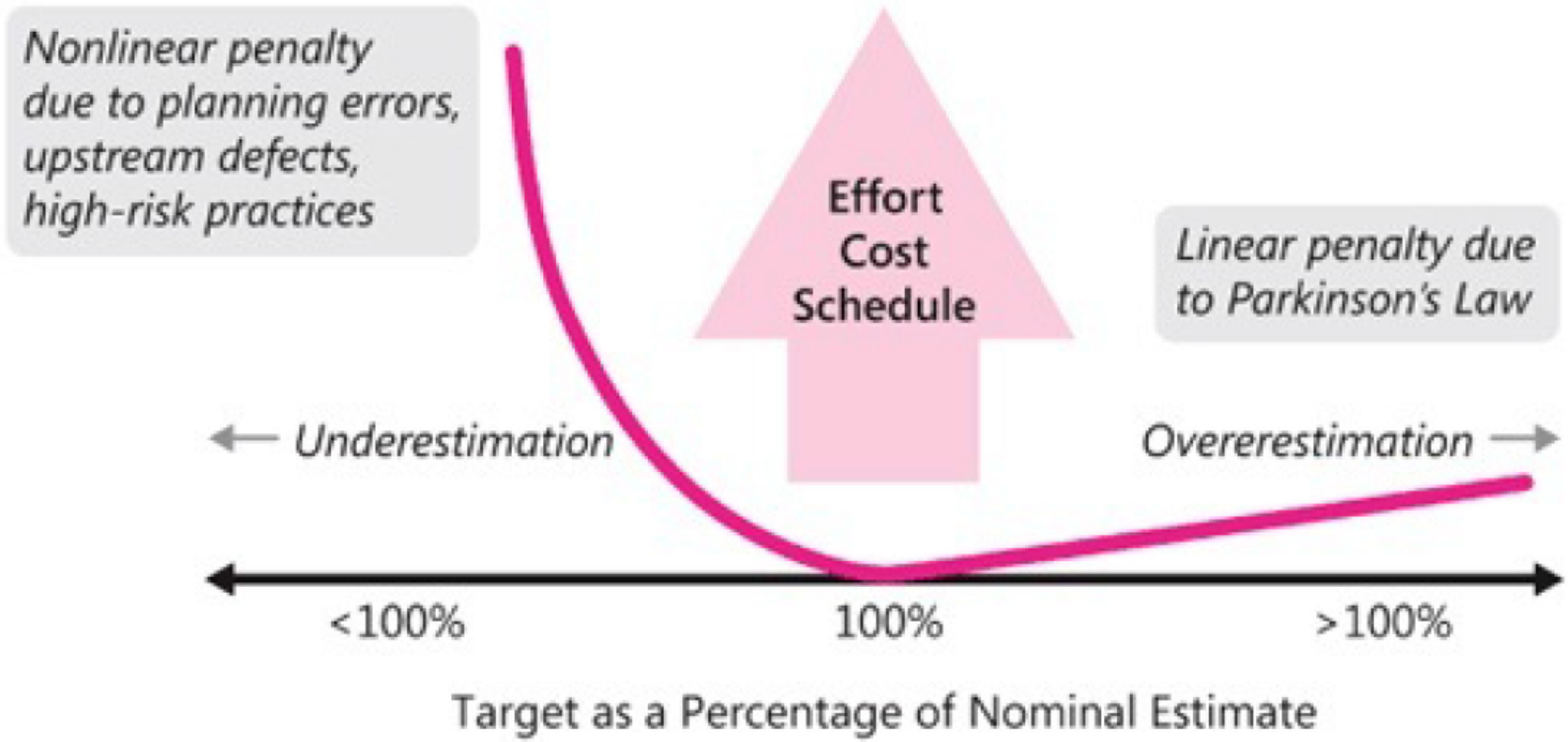
"An estimate is the most optimistic prediction that has a non-zero probability of coming true.

Accepting this definition leads irrevocably toward a method called what's-the-earliest-date-by-which-you-can't-prove-you-won't-be-finished estimating."

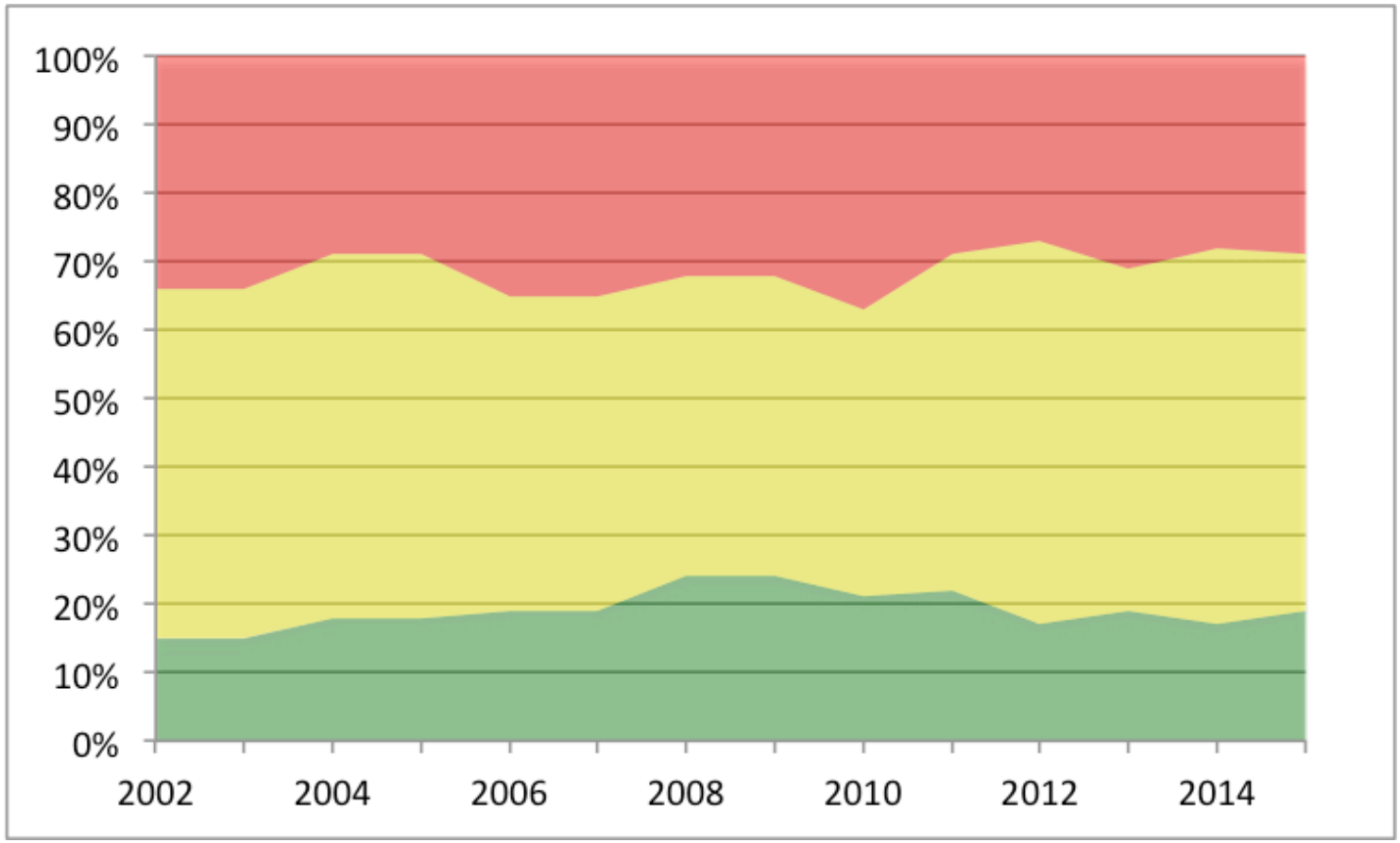
Tom DeMarco

Pricing to win means bidding as low as possible to beat the competition and win the contract.

Effects of Inaccurate Estimates

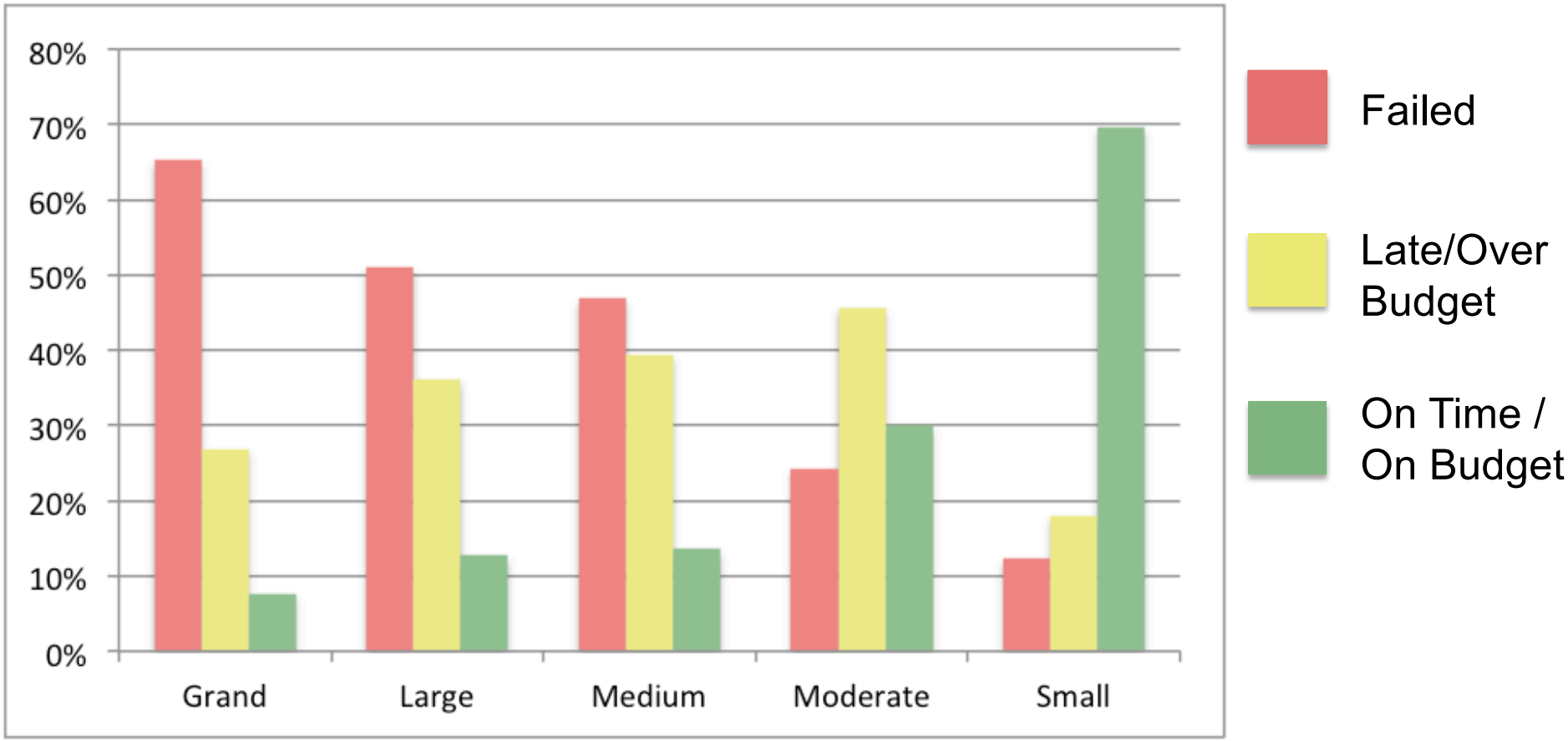


Industry's Track Record



The Standish Group, The CHAOS Report, 2002-2015

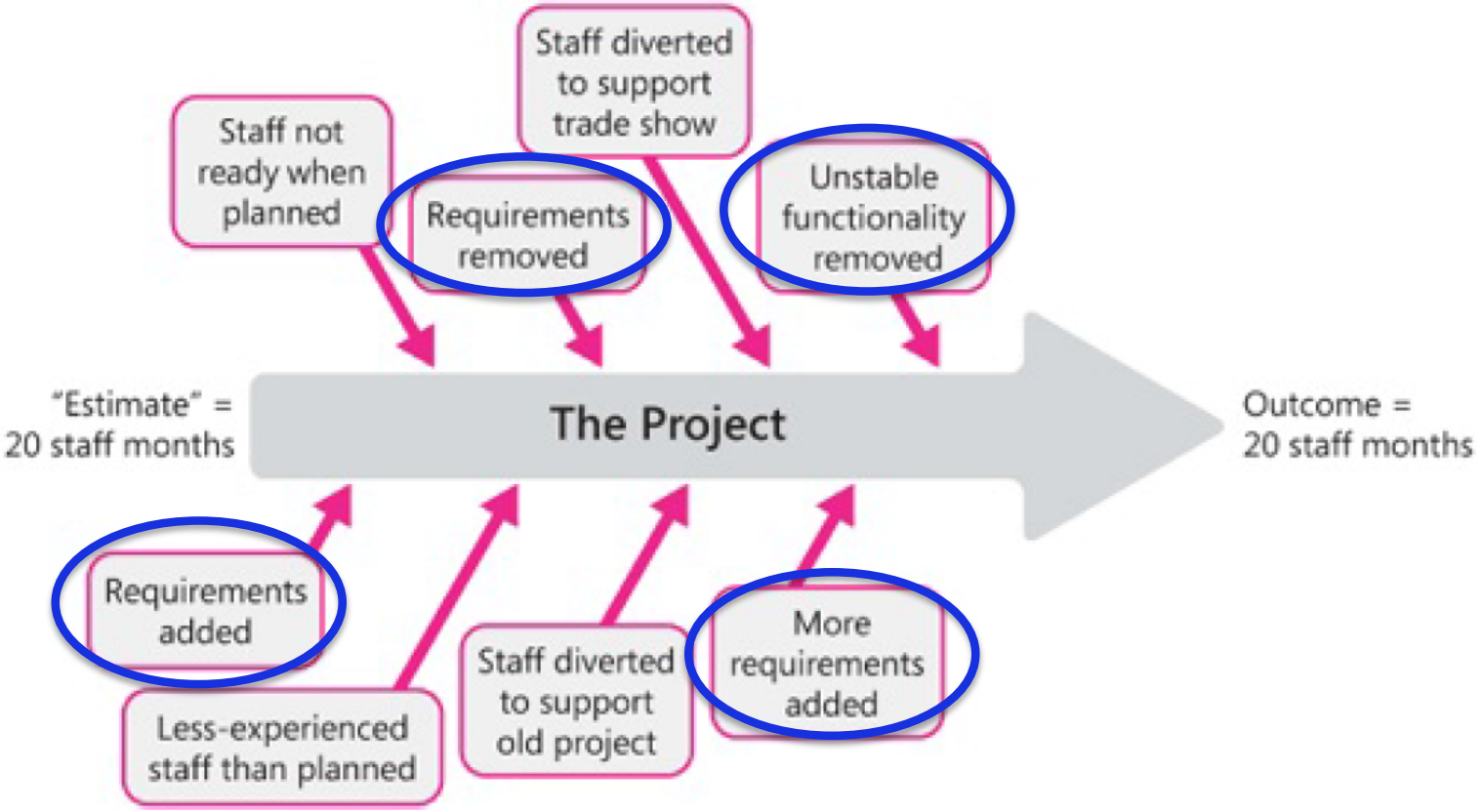
Industry's Track Record



The Standish Group, The CHAOS Report, 2002-2015

What is a Good Estimate?

A project is successful if it is delivered on time, within budget, with acceptable functionality.



Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Sources of Estimation Error

Omitted Activities

- Omitted Functional / Quality Requirements
  - Initialization, data conversion, glue code
- Software Development Activities
  - Integration, test data, performance tuning, technical reviews
- Non-Software Development Activities
  - Vacations, sick days, training, company meetings

Sources of Estimation Error

Optimism

- Developer estimates tended to contain an optimism factor of 20% to 30%
- Managers believe that projects can be completed 30% faster than previous projects
  - Team more productive, less will go wrong, no learning curve

Bias

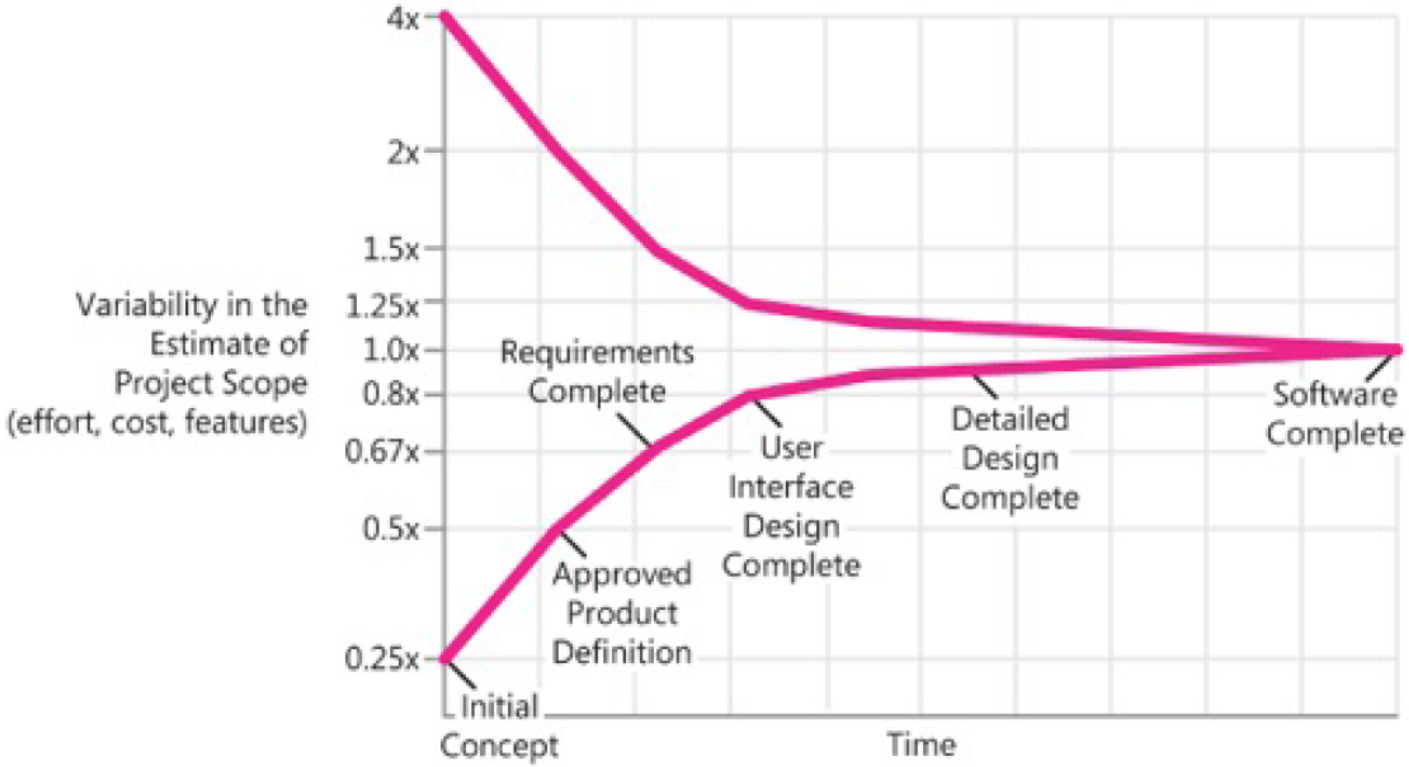
- Managers who want estimates to align with targets will put pressure on estimates, schedules, project teams.

Subjectivity

- Some estimation techniques include multiple adjustment knobs that allow subjectivity to creep in.

Estimation Error

Uncertainty about how numerous requirements details and design decisions will be made.



Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Estimation accuracy improves rapidly for the first 30% of the project.

Biggest Influences on Estimates

Project Size

- The largest driver in a software estimate is the size of the software being built
- Software projects have diseconomies of scale: larger projects require more coordination and communication

Kind of Software

- Business applications, embedded systems, real-time

Personnel Factors

- Productivity of individuals can vary by a factor of 10 or more
- Mostly accounted for if estimates are based on team's historical performance

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Data to Collect

Whenever possible count or compute your estimate rather than using expert judgment.

In order to compute software estimates based on historical data, we need to be collecting data about our projects.

- Project size
  - lines of code (LOC)(KLOC)
  - requirements
    - e.g., function points, scenarios, stories, UI screens
- Effort (staff months)
- Time (calendar months)

Collecting data on size and effort every 1 to 2 weeks can provide valuable insight into your project's dynamics.

Size Estimates in LOC

LOC measure is the most common size measure used for estimation.

Advantages:

- Easily to collect from past projects.
- There exists lots of historical data in organizations
- It allows for cross-project comparisons and estimation of future projects based on data from past projects.
- Normally, the effort and schedule estimates is based on LOC in commercial estimation tools

Size Estimates in LOC

Disadvantages:

- Different kinds of software present differences in coding rates.
- Some programmers write more efficient code.
- Lines of code are difficult to estimate directly, and must be estimated by proxy.

Estimates Based on Past Data

Compute estimates of new development tasks based on the actual work to complete past tasks.

- Project data - data generated earlier in the same project that's being estimated
- Historical data - data from the organization that will conduct the project being estimated
- Industry data - data from other organizations that develop the same basic kind of software as the software that's being estimated

Project Size - LOC

You will need to define how to count several issues:

- All source code vs only source code that's included in the released software
- Source code that is reused (i.e.,from previous versions)
- Open source code
- Third-party library code
- Blank lines and comments
- Class interfaces
- Data declarations
- line of source code that is broken across multiple lines

Estimation by Analogy

Compute a size estimate based on a piece-by-piece count of analogous elements, and the past sizes of those elements.

Subsystem	Old Project (Size)	New Project (Est. Size)	Multiplier	Old Project (LOC)	New Project (LOC)
Use Cases	10 use cases	14 use cases	1.4	5,000	7000
User Interface	14 screens	16 screens	1.1	14,000	15,400
Graphs	10 graphs	16 graphs	1.6	9,000	25,600
Reports	3 reports	5 reports	1.7	3,000	5,100
Business Rules	???	???	1.5 (?)	11,000	16,500
TOTAL				**42,000**	**69,600**

Historical data
Estimates
Computed values

Expert estimators: decomposing the problem into pieces, estimating each piece by analogy (calibrated with historical data), arrive at estimates that have a standard deviation of roughly 7%

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Function Point Analysis

- Estimations by analogy work only if the new projects are analogous to previous projects

- Function Point Analys is an attempt to ”count” the complexities of a problem, based on the details known at requirements time, and based the estimate on that count.

Function Point Analysis

Goal: More precise estimates based on product complexities (known at requirements)

We break the problem down into three parts:

1. Estimate the number function points from the requirements
2. Estimate code size from function points
3. Estimate resources required (time, personnel, money) from code size

Function Points

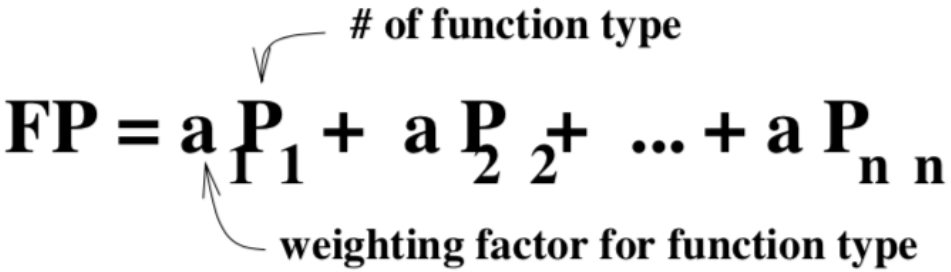
A function point is a size metric of functionality. It relates to the requirements of the software under development.

The number of function points in a system is based on the number and complexity of the following

- **External Inputs** – screens, forms, dialog boxes, messages
- **External Outputs** – screens, reports, graphs, messages
- **External Queries** – responses to input requests that do not change system data
- **Internal Logical Files** – major internal data stores (end-user data, control information)
- **External Interfaces / APIs** – files or APIs controlled by adjacent systems

Function Point Counting

A function point count is computed by adding up inputs and outputs, weighted by complexity multipliers.



Program Characteristic	Low Complexity	Medium Complexity	High Complexity
External Inputs	___ x 3	___ x 4	___ x 6
External Outputs	___ x 4	___ x 5	___ x 7
External Queries	___ x 3	___ x 4	___ x 6
Internal Files	___ x 4	___ x 10	___ x 15
External Interfaces / APIs	___ x 5	___ x 7	___ x 10

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Estimating Code Size from FPs

There are tables that list for each programming language, the number of statements in it that are required to implement one function point.

It was observed a long time ago that a programmer's productivity in terms of debugged, documented, lines of code per day is constant and is independent of the language he or she is using.

http://www.qsm.com/resources/function-point-languages-table

Estimating Code Size from FPs

Language	Average	Median	Low	High
ASP	51	54	15	69
C	97	99	39	333
C++	50	53	25	80
C#	54	59	29	70
Excel	209	191	131	315
HTML	34	40	14	48
J2EE	46	49	15	67
Java	53	53	14	134
JavaScript	47	53	31	63
.NET	57	60	53	60
Oracle	37	40	17	60
SAS	38	37	22	55
SQL	21	21	13	37

http://www.qsm.com/resources/function-point-languages-table

Structured Expert Judgment

Use expert judgment as a last resort, or for a second or third estimation.

- People doing the work will create the most accurate estimates
- It's best to decompose estimation – and estimate tasks that require no more than 2 days of effort
- It's best to estimate a range (best case vs. worst case) of sizes
  - Single-point estimates tend to be much closer to best-case estimates
- Program Evaluation and Review Technique (PERT) Formula

ExpectedCase = [BestCase + (4*MostLikelyCase) + WorstCase] /6

Commercial Estimation Tools

Commercial tools embed computationally intensive estimation methods that cannot be done easily by hand or with a calculator.

- Simulating project outcomes (with respect to probability distributions for variability in system size, productivity)
- Probability analysis of estimates
- Complex effort estimations that account for diseconomies of scale

Tools tend to have large databases of industry-average data, but they can be calibrated with your own historical data.

Construx Estimate: www.construx.com/Resources/Construx_Estimate

Summary of Size Estimation Tech.

Technique	Kinds of Sizes that can be Estimated
Analogy	features, function points, screens, GUI components, LOC
Estimation Tools	function points, LOC
Function Point Analysis	function points, LOC
Structured Judgment	features, user stories, requirements, use cases, function points, Web pages, GUI components, database tables, interface definitions, classes, functions/subroutines, LOC
GUI Elements	function points, LOC
Standard Components	function points, LOC
Story Points	story points, LOC
Wideband Delphi	features, user stories, requirements, use cases, function points, Web pages, GUI components, database tables, interface definitions, classes, functions/subroutines, LOC

It's best to create Best-Case, Worst-Case, and Expected-Case estimates.

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Effort Measures

- Time in hours vs days vs others. How many?
- Unpaid overtime vs paid overtime
- Holidays, vacation, and training
- What kinds of effort do we count? Testing? First-level management? Documentation? Requirements? Design? Research?
- How we can count the time spent supporting previous releases or the time spent supporting sales calls, trade shows, and so on?

Again, the main goal here is to define the data you're collecting well enough so that you know what you're estimating

Estimating Effort from Past Projects

Informal Comparison to Past Projects – can compute an early estimate of effort (range of values) based on the effort results from analogous past projects

- Effort = PastEffort x (Size / PastSize)
- Linear model is OK, if relevant historical data is within a narrow size range

Project	Size (LOC)	Schedule (Calendar Months)	Effort (Staff Months)	Productivity (LOC/Staff Month)
Project A	33,842	8.2	21	1,612
Project B	97,614	12.5	99	986
Project C	7,444	4.7	2	3,722
Project D	54,322	11.3	40	1,358
Project E	340,343	24.0	533	639

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Estimating Effort

What Kinds of Effort Are Included in This Estimate?

We are using historical data to estimate staff months, so this estimate includes whatever effort is included in the historical data. For example:

1. Effort for development and testing, and only for the part of the project from end of requirements through system testing.
2. Effort for requirements, project management, and user documentation.

It's common that these estimates (based on industry-average data) include all technical work, but not management work. Also they include all development work but not requirements.

Estimating Effort from ISBSG

Formulae: If you don't have your own historical data, you can compute a rough estimate of effort based on models of industry-average efforts.

International Software Benchmarking Standards Group (ISBSG) Method

Desktop Projects:

$$StaffMonths = 0.157 \times FunctionPoints^{0.591} \times MaxTeamSize^{0.810}$$

Enhancement:

$$StaffMonths = 0.669 \times FunctionPoints^{0.338} \times MaxTeamSize^{0.758}$$

New Development:

$$StaffMonths = 0.520 \times FunctionPoints^{0.385} \times MaxTeamSize^{0.866}$$

Third Generation Language Project:

$$StaffMonths = 0.425 \times FunctionPoints^{0.488} \times MaxTeamSize^{0.697}$$

Fourth Generation Language Project:

$$StaffMonths = 0.317 \times FunctionPoints^{0.472} \times MaxTeamSize^{0.784}$$

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Estimating Effort from Industry Data

If you have no historical data, there are data on industry-average effort per LOC.

Business Systems (Source Lines of Code benchmarks)

Size: New & Modified SLOC	Duration (Months)	Effort (PM)	Average Staff (FTE)	SLOC/PM
2,500	5.9	11.3	1.9	430.0
10,000	7.1	26.1	3.5	650.0
25,000	8.5	45.8	5.2	876.0
50,000	9.5	70.0	7.2	1,088.0
100,000	10.4	102.7	9.3	1,300.0
Min: 600	*1.6*	*1.63*	*0.9*	*38.5*
Max: 7,920,000	*42.2*	*2,219.65*	*202.8*	*12,403.4*

The Business Systems group includes 450 Business (IT) Systems projects completed between 2008 and 2011.

Quantitative Software Management (http://www.qsm.com/resources/performance-benchmark-tables)

Estimating Effort from Industry Data

Engineering Systems

Size: New & Modified SLOC	Duration (Months)	Effort (PM)	Average Staff (FTE)	SLOC/PM
2,500	6.7	22.0	3.2	192.2
10,000	9.7	53.0	5.4	294.5
25,000	12.0	92.4	7.1	394.0
50,000	14.2	143.0	9.3	497.3
100,000	16.8	225.7	12.2	621.0
300,000	23.8	453.4	19.3	887.7
Min: 32	*1.8*	*0.83*	*< 1*	*7.1*
Max: 2,573,612	*55.0*	*10,037.00*	*339.6*	*13,514.9*

The Engineering Systems group includes over 300 Command & Control, System Software, Telecommunications, Scientific, and Process Control projects completed on or after 2000.

Quantitative Software Management (http://www.qsm.com/resources/performance-benchmark-tables)

Estimating Effort from Industry Data

Real Time Systems

Size: New & Modified Code	Duration (Months)	Effort (PM)	Average Staff (FTE)	SLOC/ PM
2,500	9.6	12.30	1.4	211.8
10,000	13.6	55.1	4.0	223.0
25,000	17.2	143.0	8.3	244.1
50,000	20.6	281.0	14.3	250.5
100,000	25.1	596.9	23.6	259.3
300,000	33.7	1,850.3	54.4	274.3
Min: 344	*4.5*	*2.04*	*< 1*	*21.0*
Max: 2,141,000	*94.1*	*43,221.28*	*760.9*	*4,598.7*

The Real Time Systems group includes approximately 145 Avionics, Real Time, and Microcode & Firmware projects completed after 1990.

Quantitative Software Management (http://www.qsm.com/resources/performance-benchmark-tables)

Estimating Schedule from Basic Schedule Equation

There is a Basic Schedule Equation that produces good-enough estimates of schedule early in a medium-to-large sized project.

$$\text{ScheduleInMonths} = 3.0 \times \text{StaffMonths}^{1/3}$$

(where multiplier ranges from 2.0 to 4.0).

Estimating Schedule from Historical Data

If you have historical data, you can compute a schedule estimate based on the schedules from similar past projects

Medium-to-large projects (more than 50 staff months)

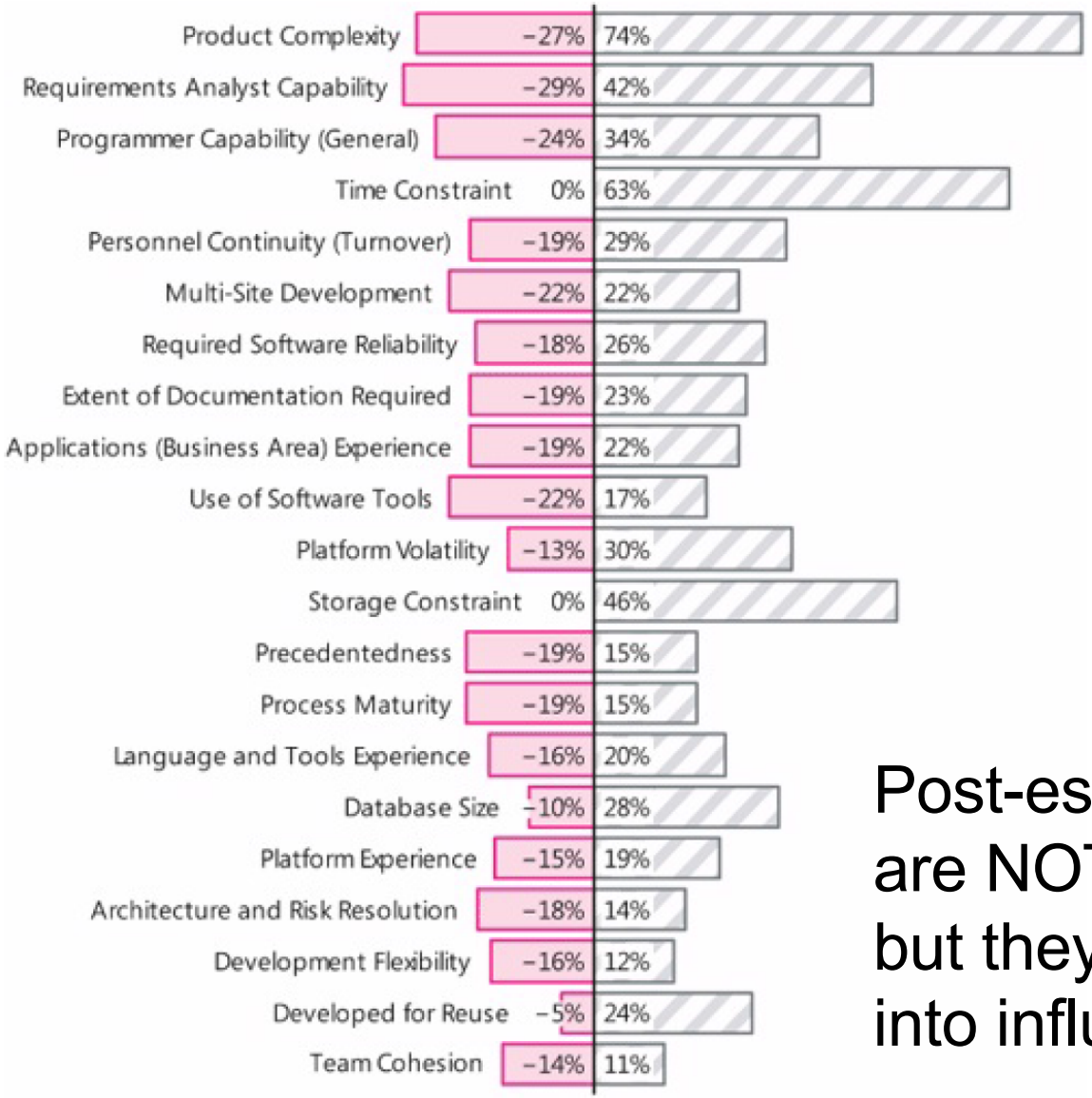
$$ScheduleInMonths = PastSchedule \times (EstimatedEffort / PastEffort)^{1/3}$$

Small projects

$$ScheduleInMonths = PastSchedule \times (EstimatedEffort / PastEffort)^{1/2}$$

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

COCOMO Adjustment Factors



Post-estimate adjustment factors are NOT RECOMMENDED, but they provide valuable insight into influences on estimations.

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Table 5-4. Cocomo II Adjustment Factors

	Ratings						
Factor	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**	**Influence**
Applications (Business Area) Experience	1.22	1.10	1.00	0.88	0.81		1.51
Database Size		0.90	1.00	1.14	1.28		1.42
Developed for Reuse		0.95	1.00	1.07	1.15	1.24	1.31
Extent of Documentation Required	0.81	0.91	1.00	1.11	1.23		1.52
Language and Tools Experience	1.20	1.09	1.00	0.91	0.84		1.43
Multisite Development	1.22	1.09	1.00	0.93	0.86	0.78	1.56
Personnel Continuity (turnover)	1.29	1.12	1.00	0.90	0.81		1.59
Platform Experience	1.19	1.09	1.00	0.91	0.85		1.40
Platform Volatility		0.87	1.00	1.15	1.30		1.49
Product Complexity	0.73	0.87	1.00	1.17	1.34	1.74	2.38
Programmer Capability (general)	1.34	1.15	1.00	0.88	0.76		1.76
Required Software Reliability	0.82	0.92	1.00	1.10	1.26		1.54
Requirements Analyst Capability	1.42	1.19	1.00	0.85	0.71		2.00
Storage Constraint			1.00	1.05	1.17	1.46	1.46
Time Constraint			1.00	1.11	1.29	1.63	1.63
Use of Software Tools	1.17	1.09	1.00	0.90	0.78		1.50

Table 5-5. Cocomo II Adjustment Factors

Cocomo II Factor	Influence	Observation
Applications (Business Area) Experience	1.51	Teams that aren't familiar with the project's business area need significantly more time. This shouldn't be a surprise.
Architecture and Risk Resolution	1.38[*]	The more actively the project attacks risks, the lower the effort and cost will be. This is one of the few Cocomo II factors that is controllable by the project manager.
Database Size	1.42	Large, complex databases require more effort project-wide. Total influence is moderate.
Developed for Reuse	1.31	Software that is developed with the goal of later reuse can increase costs as much as 31%. This doesn't say whether the initiative actually succeeds. Industry experience has been that forward-looking reuse programs often fail.
Extent of Documentation Required	1.52	Too much documentation can negatively affect the whole project. Impact is moderately high.
Language and Tools Experience	1.43	Teams that have experience with the programming language and/or tool set work moderately more productively than teams that are climbing a learning curve. This is not a surprise.
Multi-Site Development	1.56	Projects conducted by a team spread across multiple sites around the globe will take 56% more effort than projects that are conducted by a team co-located at one facility. Projects that are conducted at multiple sites, including outsourced or offshore projects, need to take this effect seriously.

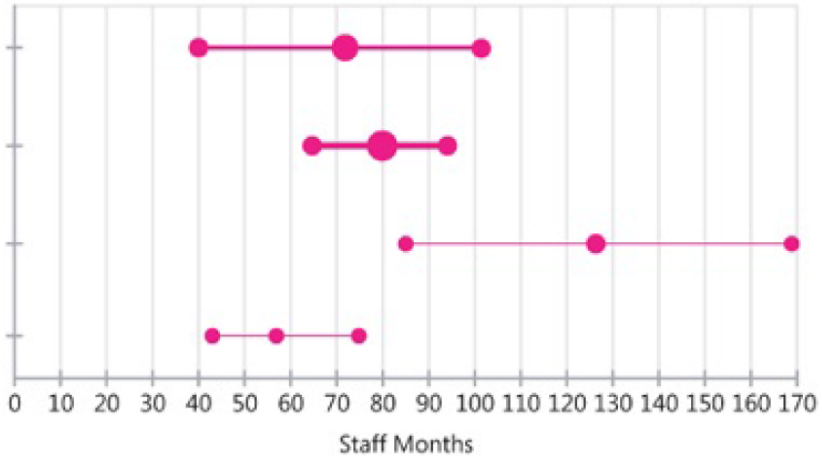
Best Practices

Decompose the estimation task into multiple subtasks and combine the results

- some estimation errors will cancel each other out

Use multiple estimation techniques and compare the results

- convergence suggests that you have good accuracy
- spread suggests some factors have been overlooked



Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Best Practices

Decompose the estimation task into multiple subtasks and combine the results

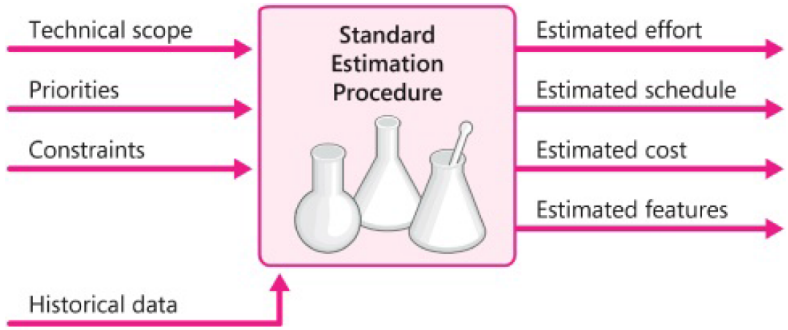
- some estimation errors will cancel each other out

Use multiple estimation techniques and compare the results

- convergence suggests that you have good accuracy
- spread suggests some factors have been overlooked

Define the estimation procedure in advance

- what granularity of task to count
- what multiple techniques to use
- when to re-estimate
- what project data to collect



Recombining Piecewise Estimates

Suppose you have a software estimate for a new project or iteration

Use Case	Best Case (staff months)	Expected Case (staff months)	Worst Case (staff months)
UC 1	4.1	5.0	6.0
UC 2	5.2	6.4	7.7
UC 3	6.0	7.1	8.2
UC 4	2.4	3.1	3.6
UC 5	5.8	7.0	8.2
TOTAL	**23.5**	**28.6**	**33.7**

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Confidence Interval

Goal:

Percentage Confident	Effort Estimate (staff months)
2	23.5
10	25.4
16	26.1
20	26.5
25	26.9
30	27.3
40	28.0
50	28.6
60	29.3
70	30.0
75	30.3
80	30.8
84	31.2
90	31.8
98	33.7

Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Confident Interval

1. Combine the standard deviations of the piecemeal estimates by summing the variances of the estimates and taking the square root.

Use Case	Best Case (BC)	Worst Case (WC)	Standard Deviation (WC-BC)/2	Variance (SD^2)
UC 1	4.1	6.0	0.95	0.902
UC 2	5.2	7.7	1.25	1.56
UC 3	6.0	8.2	1.1	1.21
UC 4	2.4	3.6	0.6	0.36
UC 5	5.8	8.2	1.2	1.44
TOTAL	**23.5**	**33.7**		**5.48**
Combined Standard Deviation				**2.34**

Confidence Interval

2. Can use a table of standard deviations to compute a percentage likelihood.

Percentage Confident	Calculation	Percentage Confident	Calculation
2	Expected Case – (2 x SD)	60	Expected Case + (0.25 x SD)
10	Expected Case – (1.28 x SD)	70	Expected Case + (0.52 x SD)
16	Expected Case – (1 x SD)	75	Expected Case + (0.67 x SD)
20	Expected Case – (0.84 x SD)	80	Expected Case + (0.84 x SD)
25	Expected Case – (0.67 x SD)	84	Expected Case + (1 x SD)
30	Expected Case – (0.52 x SD)	90	Expected Case + (1.28 x SD)
40	Expected Case – (0.25 x SD)	98	Expected Case + (2 x SD)
50	Expected Case		

Significant Digits

Percentage Confident	Effort Estimate (staff months)
25	28.6 – (0.67 x **2.34**) = ~~27.0322~~ 27
50	28.6
75	28.6 + (0.67 x **2.34**) = ~~30.1678~~ 30
90	28.6 + (1.28 x **2.34**) = ~~31.5952~~ 32

Lastly, try to report estimates in significant digits and units that are consistent with the estimate's underlying accuracy and precision

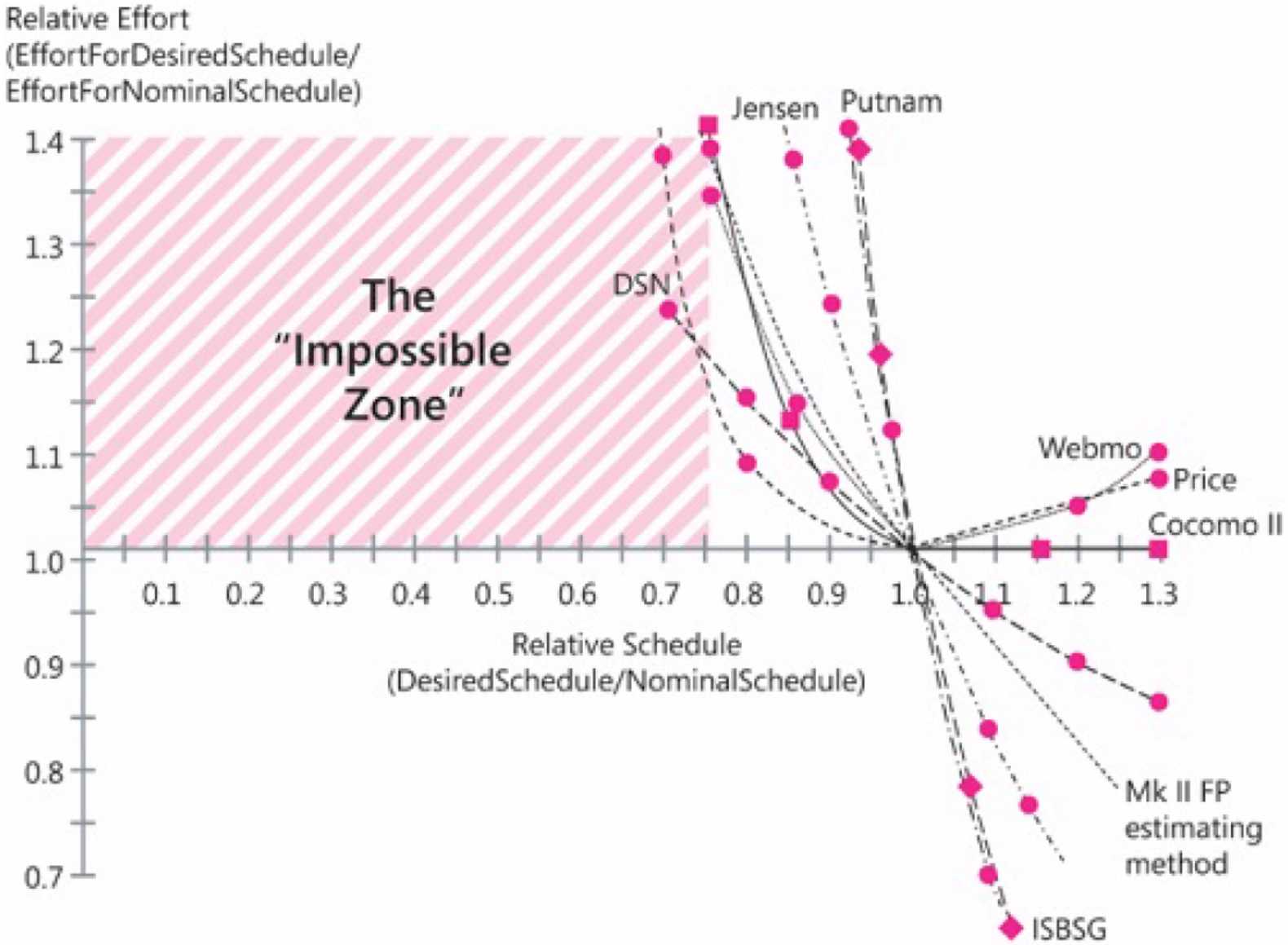
Project Control

Once we make an estimate, and make a commitment to deliver functionality and quality by a particular date, then we control the project to meet the target.

- remove noncritical requirements
- redefine requirements
- replacing less-experienced staff with more-experienced staff

Software estimation determines whether a project's targets are realistic enough to allow the project to be controlled to meet them. If the initial target and initial estimate are within about 20% of each other, the project manager should have enough maneuvering room to meet the project's business goals.

Schedule Compression



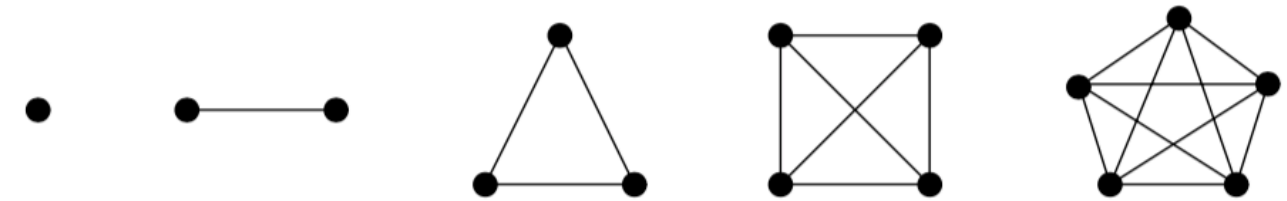
Steve McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, 2006.

Schedule Compression

Shortening the nominal schedule increases overall effort:

Famous quote from Freddy Brooks: "*Adding more people to a late project makes it even later*."

Importance of Communication in a Group Project:



1,0 2,1 3,3 4,6 5,10

number of persons, lines of communication

Schedule Compression

There is an Impossible Zone, and you can't beat it.

One woman gestates a baby in 9 months. How many months are required for three women to gestate a baby?

If 4 people can write a program in 10 months, can 40 people write the same program in one month? Can 8,000 people write it in one day?

Certain tasks require a certain minimum amount of time and throwing more personnel at the tasks does not reduce the time needed.

Schedule and Staffing Constraints

Schedule estimation techniques assume that, whatever the nominal schedule is, you'll be able to adjust your team size to fit the level indicated.

If staffing is fixed, then the only flexibility that you have is to reduce the feature set.

Summary

Software estimation is critical in providing input to decisions about project scoping, planning, and tracking.

- Estimations vs. targets vs. commitments
- Sources of inaccuracies in estimations
- Techniques for estimating project size, effort, and schedule
- Data collection to support calibration in techniques
- Best practices
- Confidence interval
- Project control