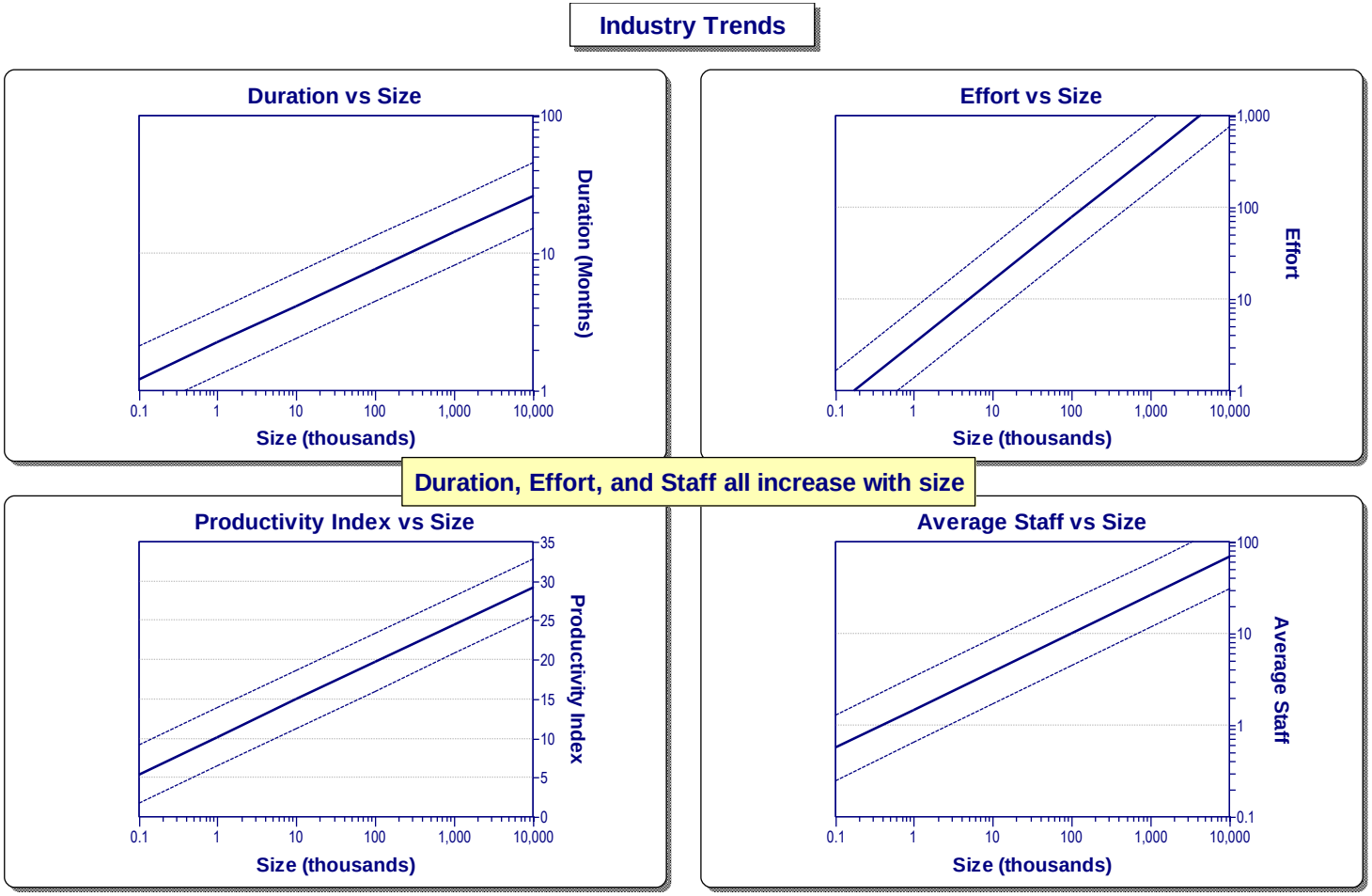
Quantitative Software Management, Inc.
2000 Corporate Ridge, Suite 900
Mclean, VA 22102
703.790.0055 • 703.749.3795 (fax)
info@qsm.com • www.qsm.com

PRACTICAL SOFTWARE ESTIMATION: PRINCIPALS AND PITFALLS

Outline

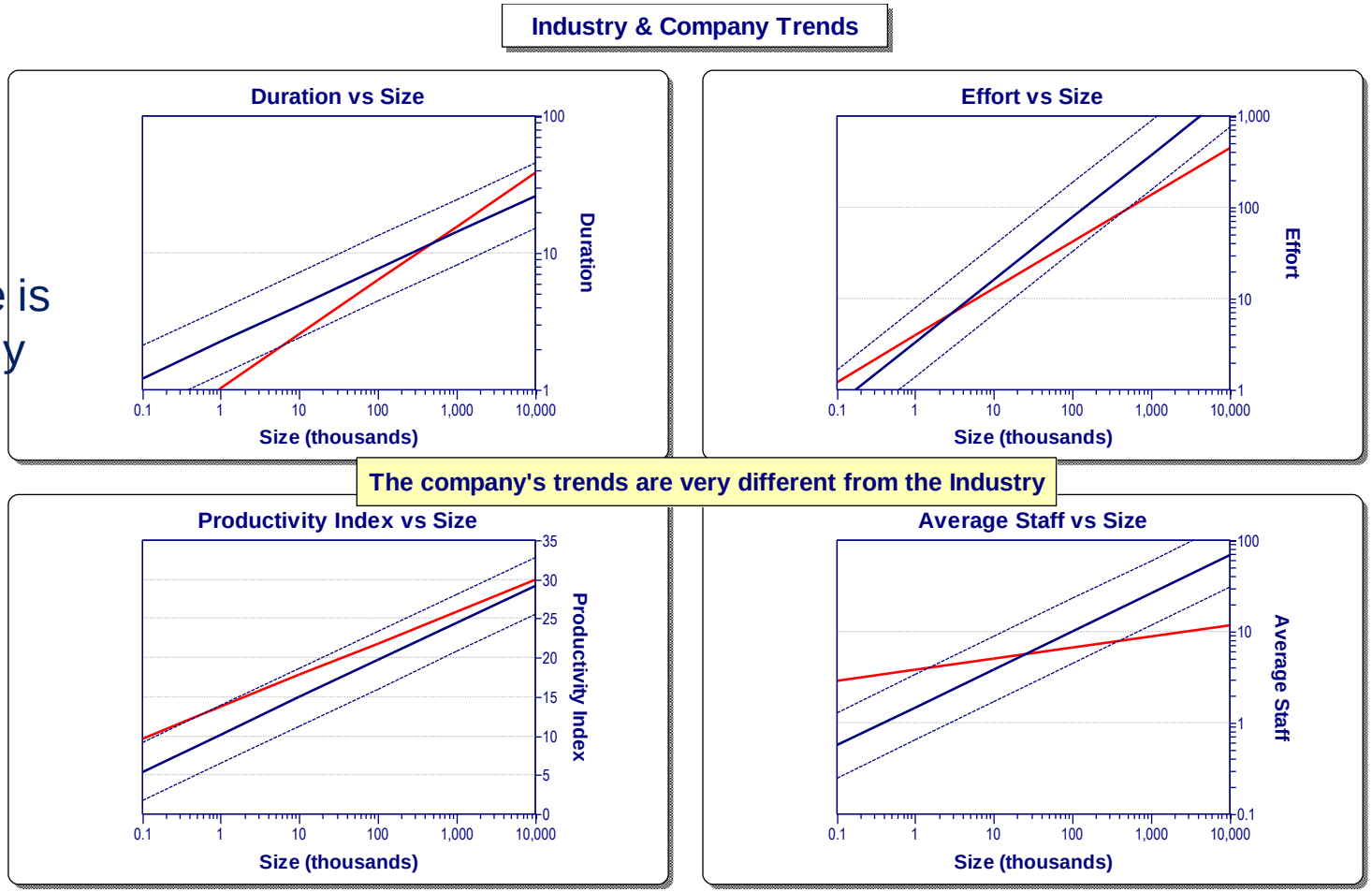
- Software size, the key component
- Non-linear relationships
  - Why?
- The application conceived is not the application delivered
- What is software development? (It may not be what you think!)
- Potential pitfalls
  - Ratio based estimating
  - Commitments based on partial knowledge
  - Estimating best case scenarios

Software Size Industry Trends



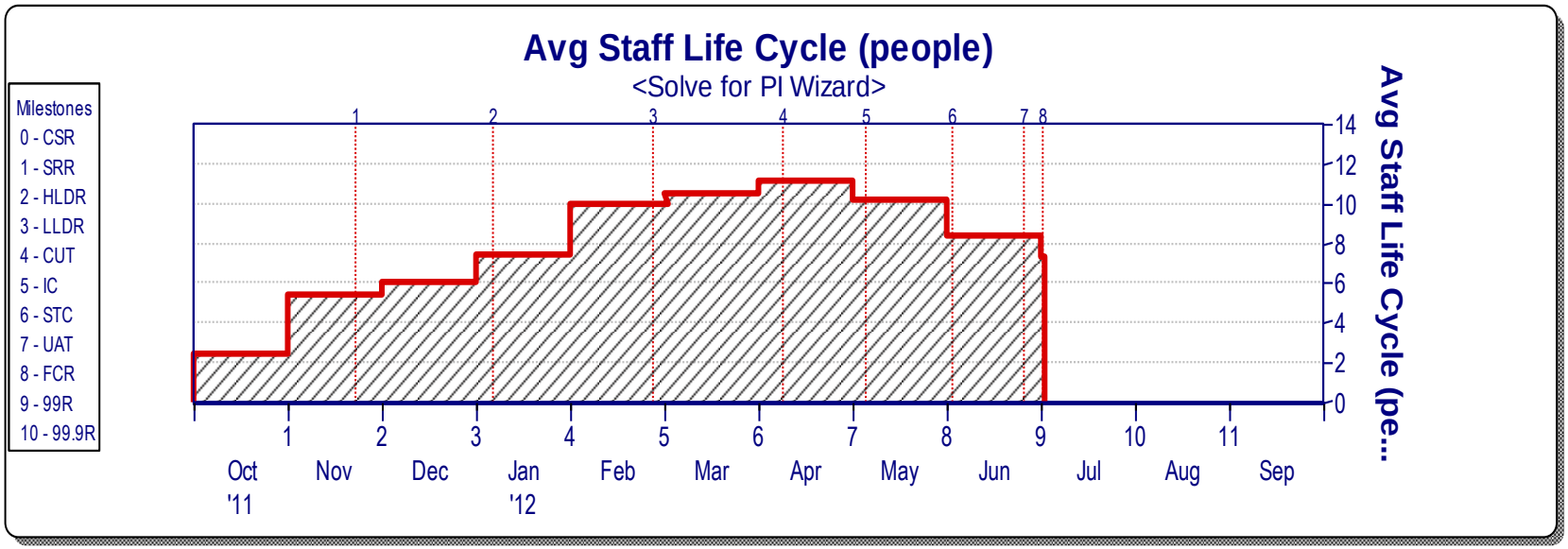
Software Size
Company vs Industry Trends

Red line is company average



Your own organizational data is superior for tuning estimates

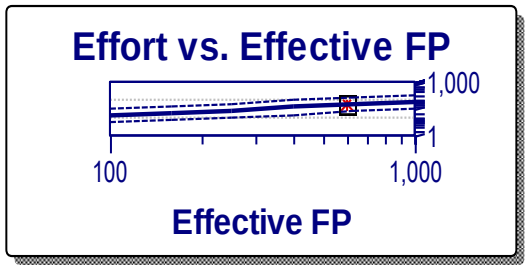
Non-linear Relationships an Optimal Estimate

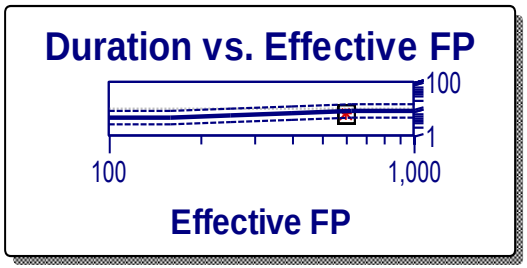


SOLUTION PANEL - <Solve for PI Wizard>

	C&T	Life Cycle	
Duration	6.5	9.0	Months
Effort	54	70	PM
Cost	1118.7	1454.0	$ (K)
Peak Staff	11.2	11.2	people
MTTD	1.737	1.737	Days

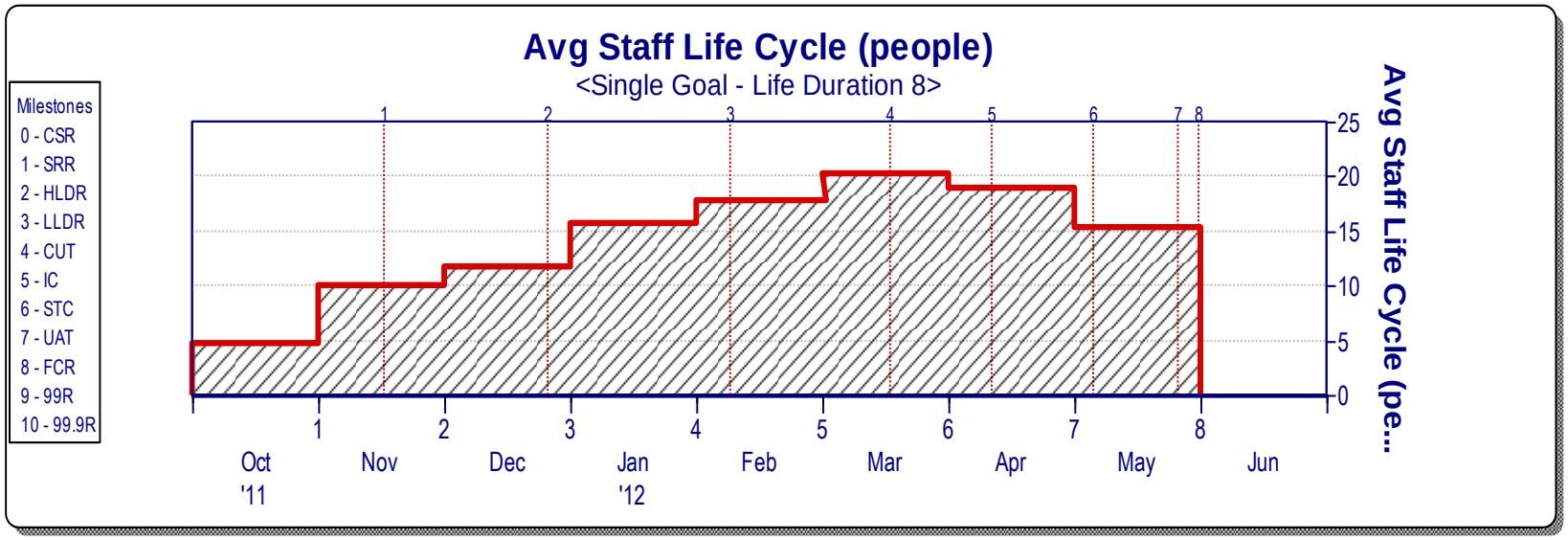
PI=15.9 MBI=4.7 Eff FP=600





This 600 FP project takes 9 months to complete and is average for effort and schedule compared to the industry IT trends

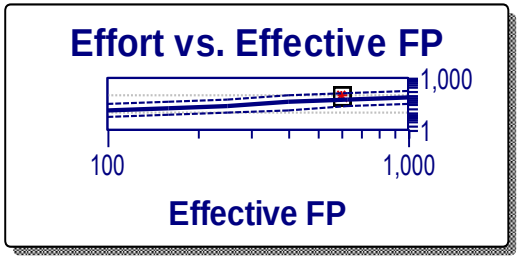
Non-linear Relationships
Schedule Compressed to 8 Months

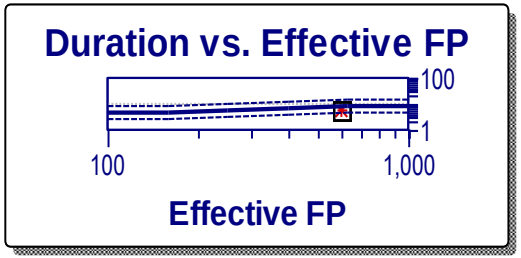


SOLUTION PANEL - <Single Goal - Life Duration 8>

	C&T	Life Cycle	
Duration	5.7	8.0	Months
Effort	87	113	PM
Cost	1805.8	2347.1	$ (K)
Peak Staff	20.4	20.4	people
MTTD	1.188	1.188	Days

PI=15.9 MBI=5.9 Eff FP=600





Project effort increased from 70 to 113 staff months (61%)with a 1 month (11%) schedule reduction

Why?

- To reduce schedule, teams use these strategies:

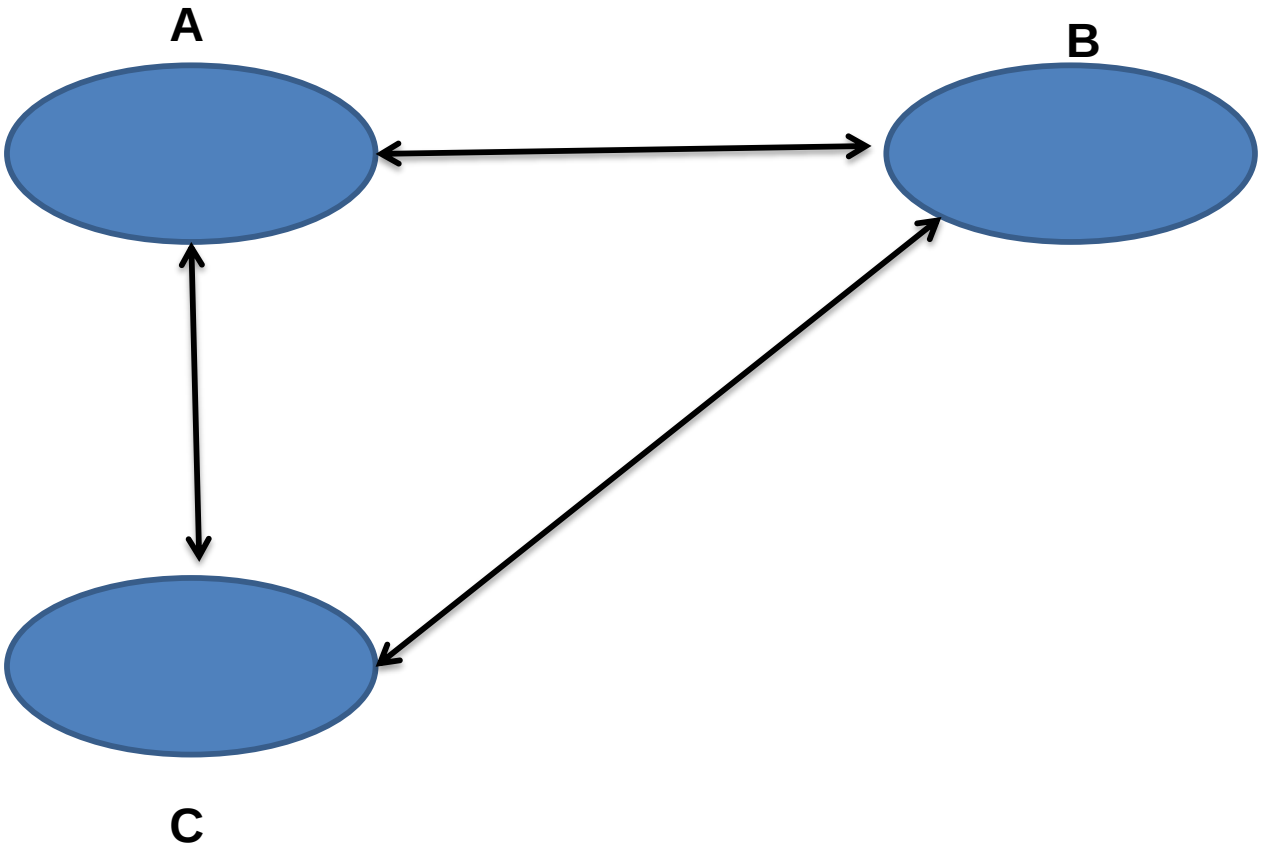
  Work overtime. Unfortunately, developers become less productive the longer they work and introduce more defects into the system which require additional time to identify and correct.

  Add staff. Three problems arise. (1) New team members require time to get up to speed. (2) Getting them up to speed decreases the productivity of the existing team members. (3) Each new team member creates many new communication channels, all of which may potentially slow down development. A 3 person team has 4 communication paths while a 4 person team has 11.

$$\textbf{Size} = \textbf{Effort}^{a} \times \textbf{Time}^{b} \times \textbf{Productivity}$$

where $a = \frac{1}{3}$ and $b = \frac{4}{3}$

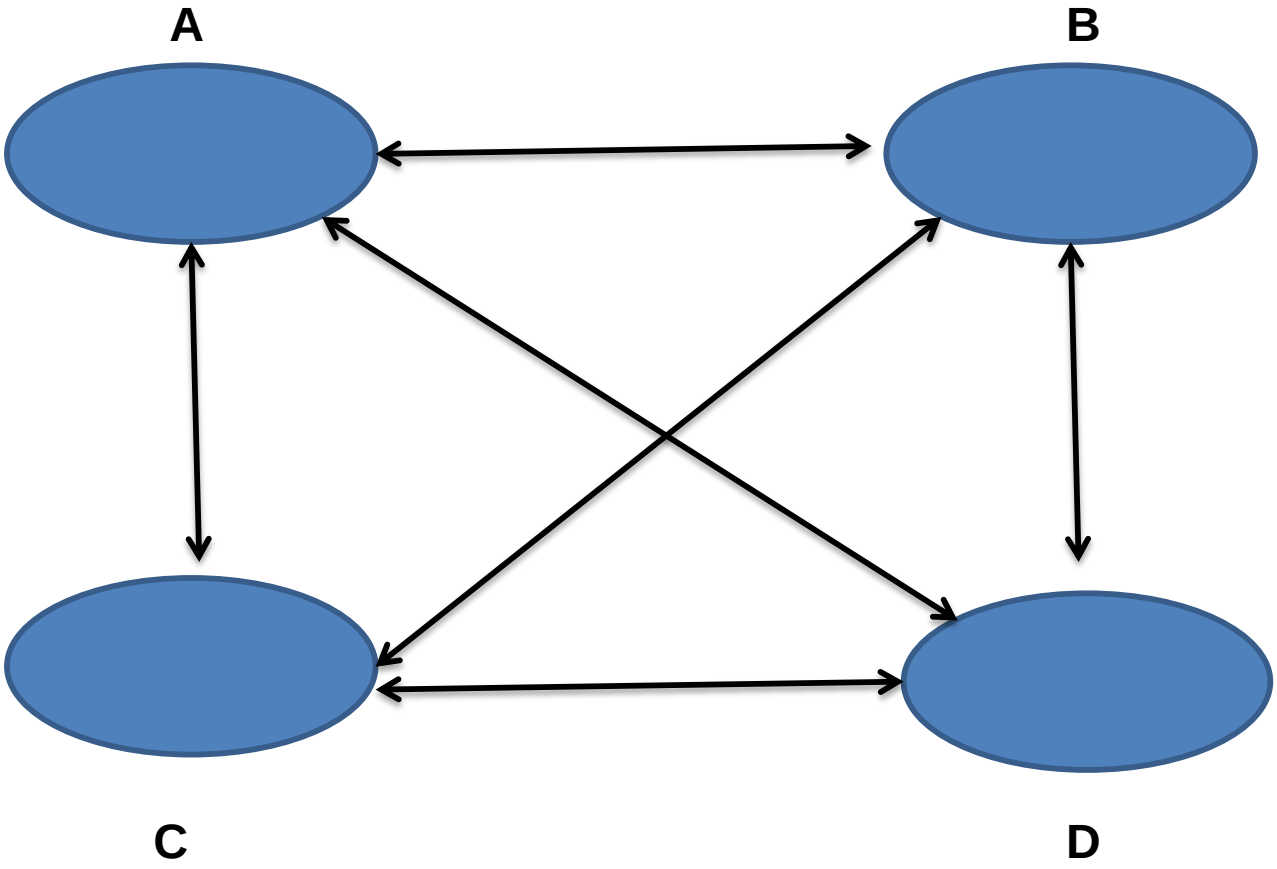
Three Person Team



Communication paths

- AB
- AC
- BC
- ABC

Four Person Team

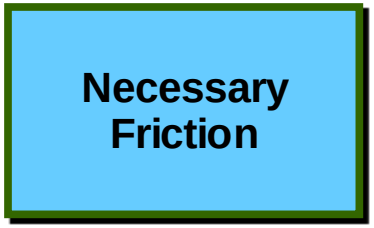


Communication paths

- AB
- AC
- BC
- ABC
- BD
- CD
- AD
- ABD
- BCD
- CAD
- ABCD

Real Work, Necessary Friction, Optional Chaos

Effort that goes directly into transcribing knowledge you already have into the system being built

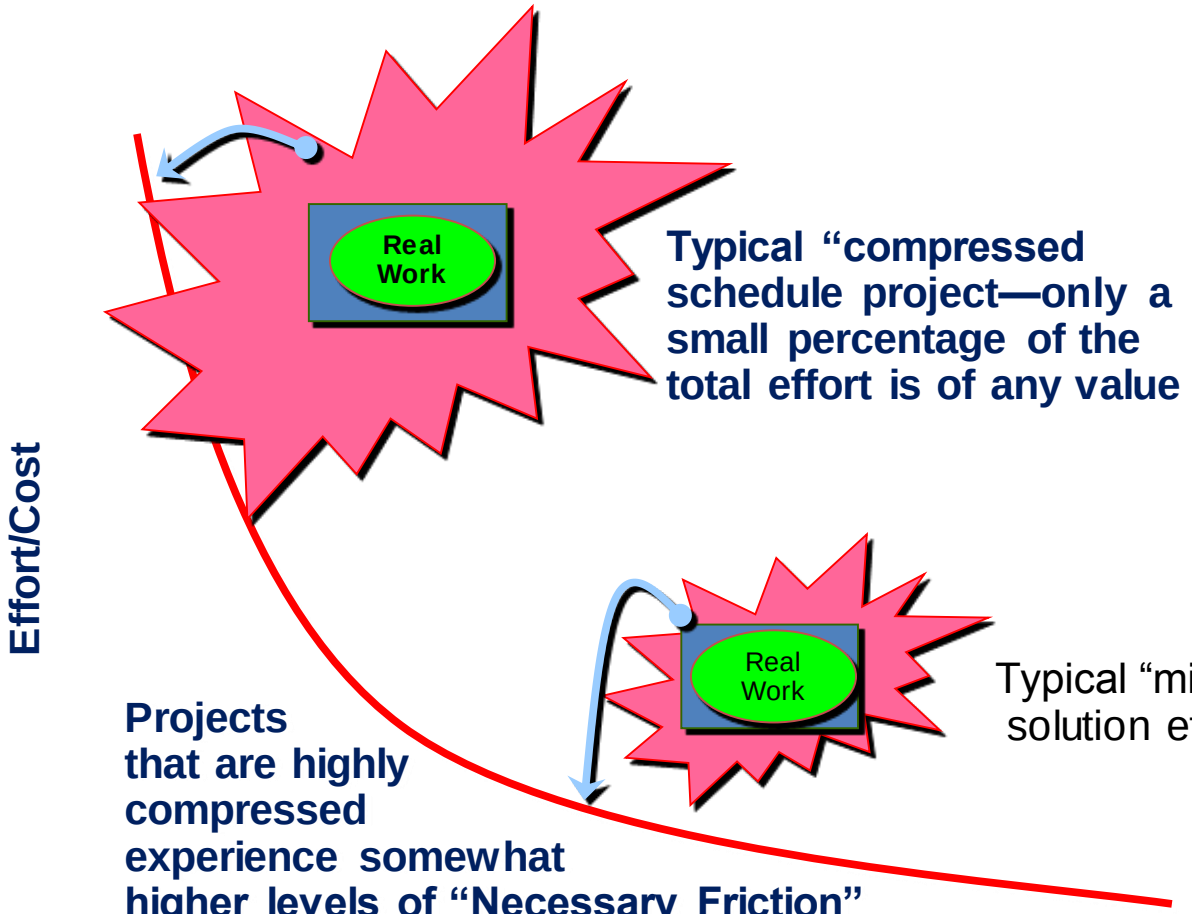


The work necessary to *discover* knowledge you do not already have

The "Brownian Motion" of the project. It is waste

Source:
Armour, Phillip G. "Real Work, Necessary Friction, Optional Chaos"
Communications of the ACM
Vol 47 No 6 June 2004

Real Work, Necessary Friction, Optional Chaos



Source:
Armour, Phillip G.
"Real Work, Necessary Friction, Optional Chaos"
Communications of the ACM
Vol 47 No 6 June 2004

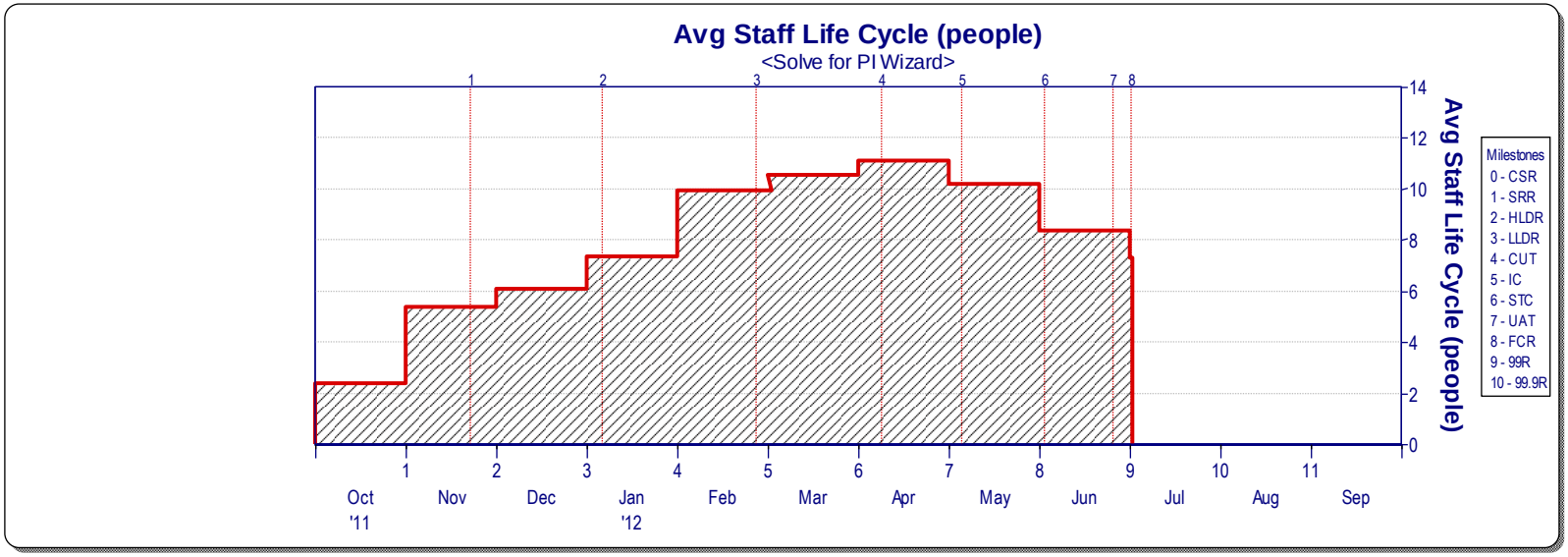
The System Conceived is not the System Delivered

- Projects grow for 2 reasons:
  - Requests for additional functionality
  - Full implications of requirements becomes known
- Schedule, Staffing, Budget usually established based on higher level requirements
  - Big pictures seem simpler than they are. Full implications not known
  - Detailed requirements not fleshed out
  - Projects contain unknown unknowns that will impact size, schedule, and effort
- Metrics can help address this issue

Accounting for Size Growth
Author's Study

- 55% projects used more effort than planned, 26% less (average of 16% more)
- 50% projects had longer schedules, 16% shorter (average 8% longer)
- 90% projects were larger than planned, 10% smaller (average 15% larger)
- Simplest accommodation is to **estimate the project 15% larger**
  - The functionality is already there. You just can't see it
- Another way to account for size growth is to estimate at a higher assurance level for cost/effort and schedule

Accounting for Size Growth



SOLUTION PANEL - <Solve for PI Wizard>

	C&T	Life Cycle	
Duration	6.5	9.0	Months
Effort	54	70	PM
Cost	1118.7	1454.0	$ (K)
Peak Staff	11.2	11.2	people
MTTD	1.737	1.737	Days
Start Date	12/18/2011	10/1/2011	

PI=15.9 MBI=4.7 Eff FP=600

Schedule = 9 months, effort = 70 person months. When duration is increased 8% and effort 16%, they become 9.72 months and 81.2 person months respectively

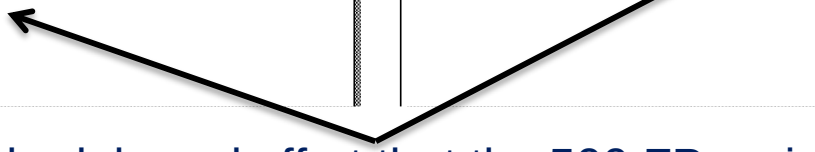
Accounting for Size Growth

Life Duration (Months) Risk Profile - Schedule compression demo
<Solve for PI Wizard>

Assurance Level (%)	Life Duration (Months)
1	7.47
5	7.93
10	8.17
15	8.34
20	8.47
25	8.58
30	8.68
35	8.77
40	8.86
45	8.95
50	9.03
55	9.11
60	9.20
65	9.29
70	9.38
75	9.48
80	9.59
85	9.72
90	9.89
95	10.13
99	10.59

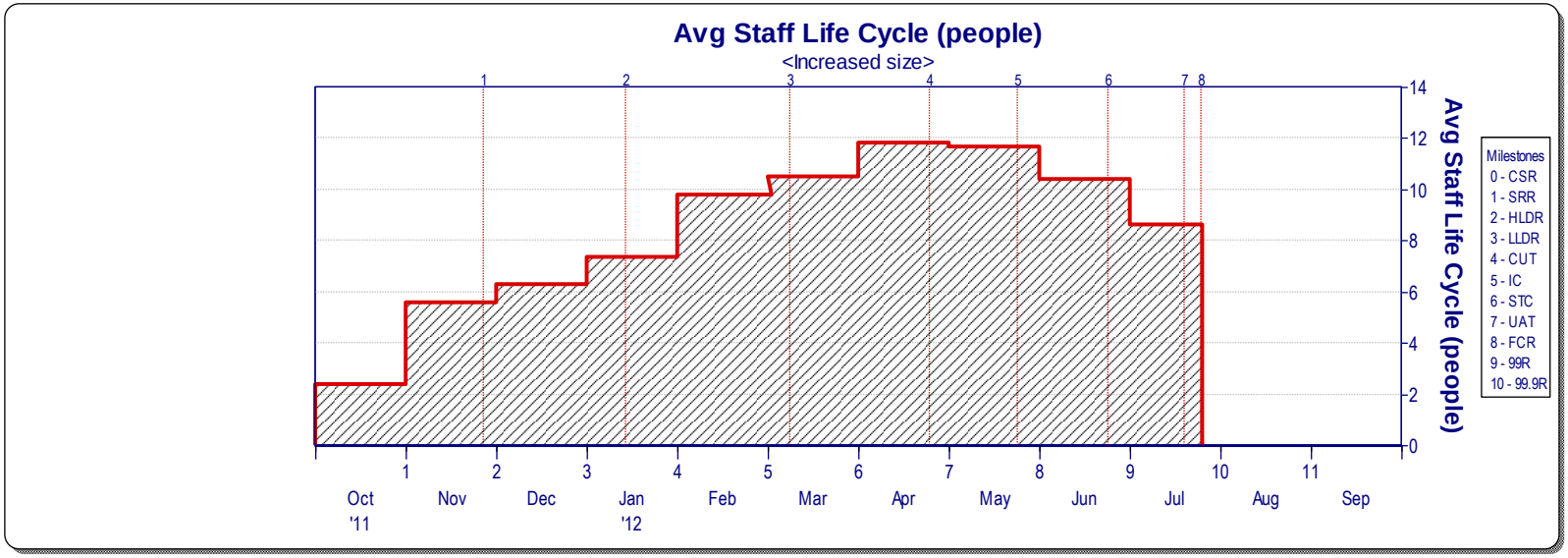
Life Effort (PM) Risk Profile - Schedule compression demo
<Solve for PI Wizard>

Assurance Level (%)	Life Effort (PM)
1	6.79
5	25.25
10	35.09
15	41.73
20	47.01
25	51.54
30	55.60
35	59.37
40	62.94
45	66.40
50	69.79
55	73.18
60	76.64
65	80.21
70	83.98
75	88.04
80	92.57
85	97.85
90	104.49
95	114.33
99	132.79



The additional schedule and effort that the 500 FP project is likely to incur can be accounted for by estimating effort to a 65 – 70% probability and schedule to an 85% probability

Accounting for Size Growth by Increasing Size



SOLUTION PANEL - <Increased size>

	C&T	Life Cycle	
Duration	7.1	9.8	Months
Effort	63	82	PM
Cost	1306.6	1698.1	$ (K)
Peak Staff	12.0	12.0	people
MTTD	1.662	1.662	Days
Start Date	12/24/2011	10/1/2011	

PI=15.9 MBI=4.4 Eff FP=690

Project size has been increased to 690 FP to account for the projected 15% size growth. Schedule = 9.8 months and effort = 82 person mths

Either increasing project size or estimating schedule and effort to higher assurance levels produce concurring estimates that account for the project growth that occurs

What is Software Development?

Things We Have Tried to Improve Software Development

- Structured programming
- Lifecycle methodologies
- CASE tools, Code generators
- 3gl, 4gl, … languages
- Object oriented programming
- Graphical user interface (GUI)
- ERP packages
- CMMI
- Service oriented architecture (SOA)
- Cloud computing
- Outsourcing
- Software as a Service (SaaS)
- Agile Development

The focus has been on **Tools, Process,** and **Architecture**

Silver Bullet Problems

- Neither individually nor in concert with others have the "silver bullets" produced more than linear improvement in productivity, quality, or time to market
- Offer technical solutions to a non-technical problem
  - Paradigm has been to transform custom artisan work into assembly line production
  - Software is not a manufacturing process. Solutions designed to improve manufacturing are not applicable to software development
  - Focusing on the people and team aspects of development appears to produce the best results

Software Development is not Manufacturing

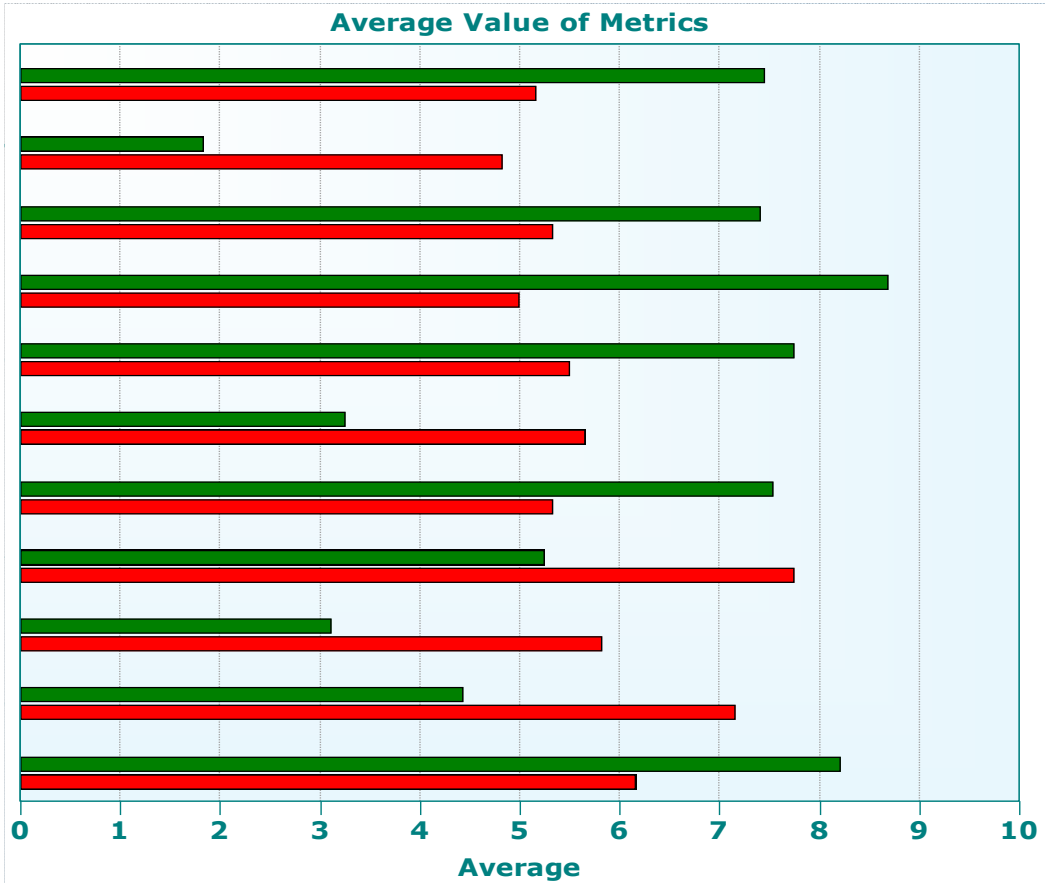
- In manufacturing
  - We know what to do and how to do it
  - We automate the process
  - If we want to double the output, we add another line or a second shift or increase machine speed
- In software development
  - To some extent we don't know what to do nor how to do it
  - We spend most of out time not building the system, but figuring out what to build and how to build it
- Software development is a knowledge acquisition process with software as the tangible output of this process

Differentiators
High and Low Performing Projects

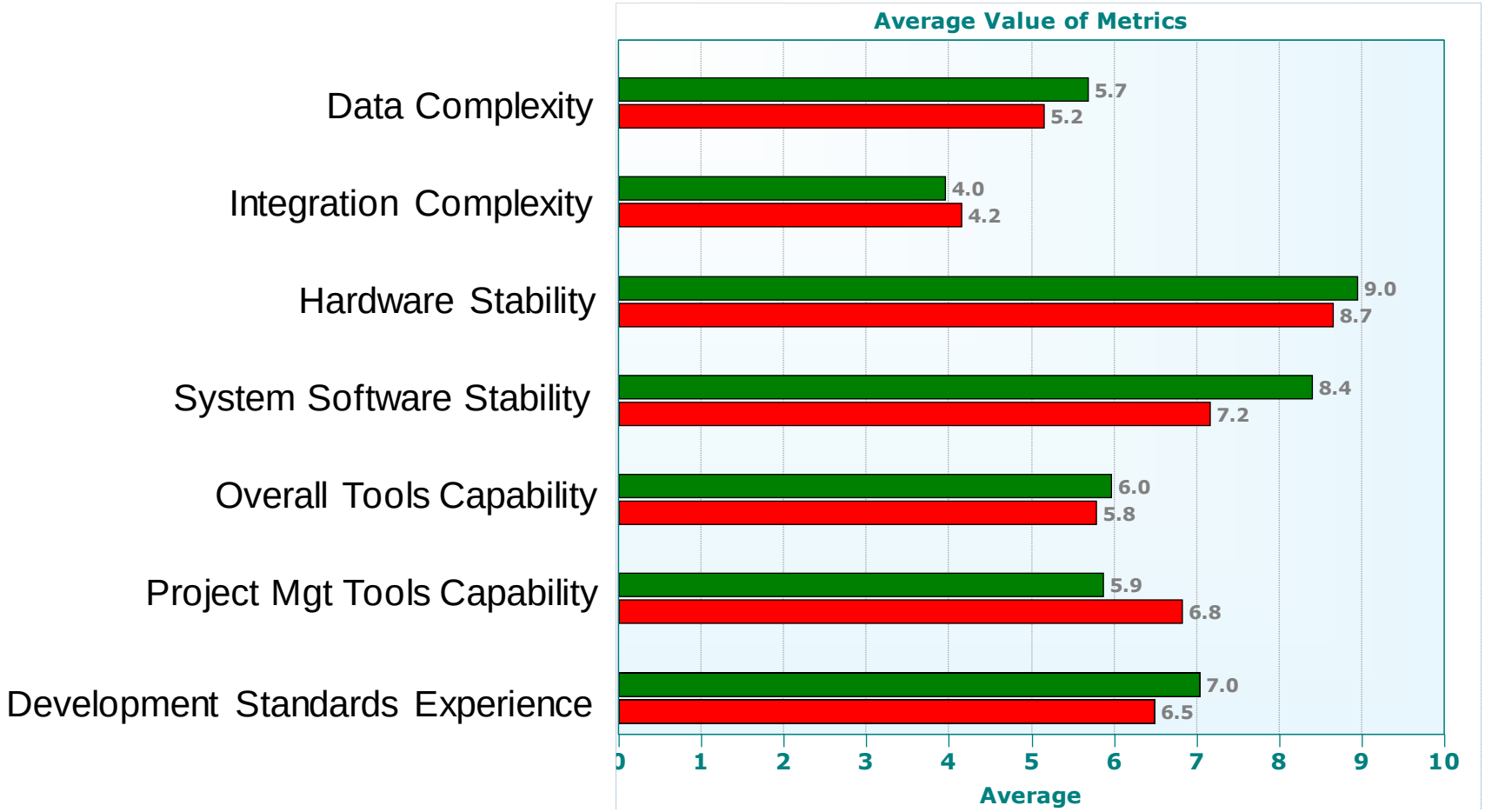
People, Communication, Knowledge
- Management Efficiency
- Staff Turnover
- Team Skill
- Motivation
- Cohesiveness
- Communication
- Knowledge

Complexity
- Overall Difficulty
- Customer Interface
- Documentation Rqmts

Tools
- Programming Tools Capability



Non-Differentiators



Potential Pitfalls
Ratio Based Estimating

Project A: 20 FP per staff month

Includes schedule & effort for development

Project B: 15 FP per staff month

Includes schedule & effort for requirements, analysis, design, & development

Ratio based estimating does not account for:

- The impact of project schedule
- The quality of the delivered product
- The lifecycle boundaries used to collect effort

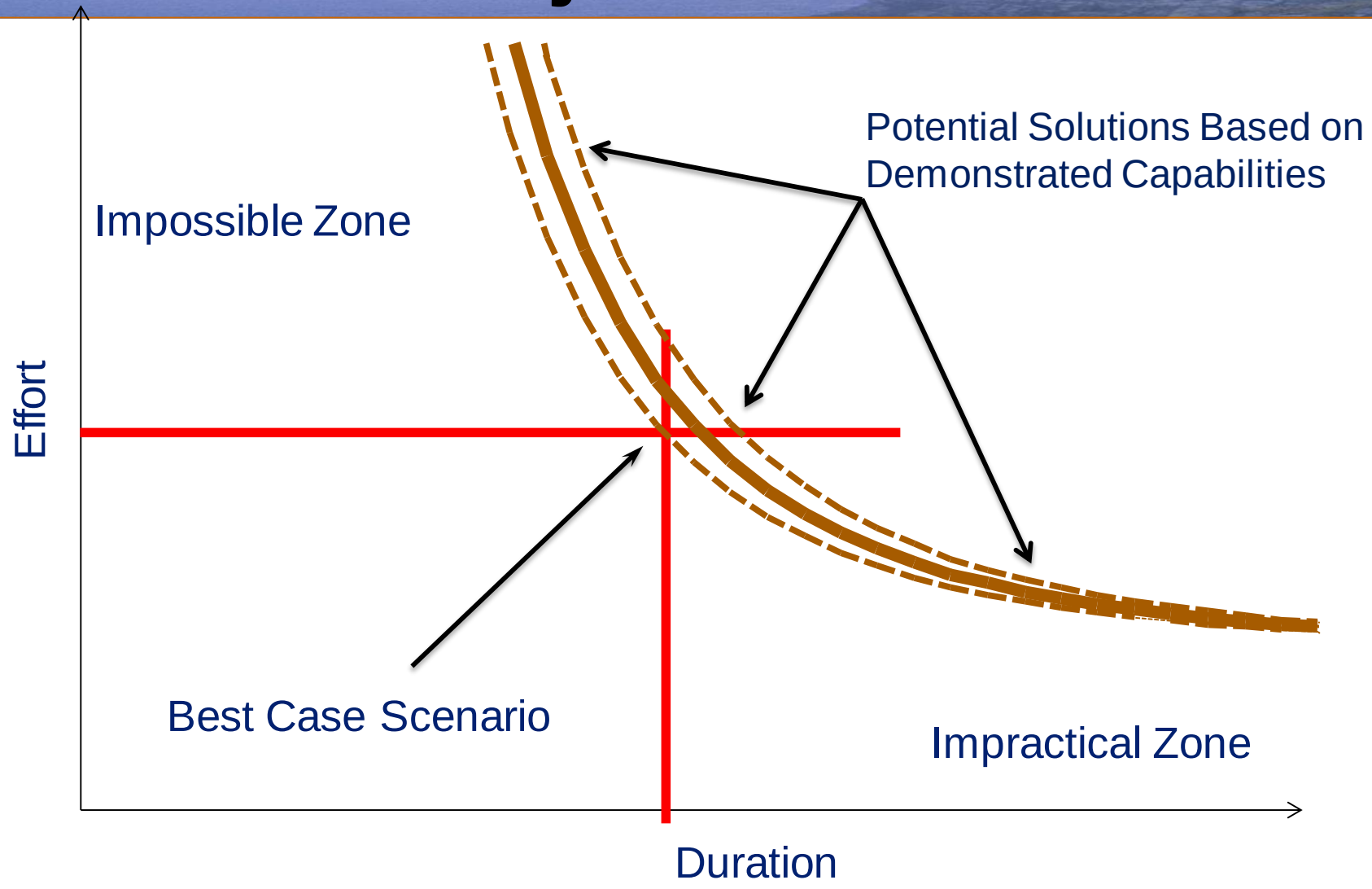
Commitments Based on Partial (Incomplete) Knowledge

- Systems are always more complex than they appear from a high level view
  - Contain more functionality
  - Have problems that must be discovered and resolved
- Estimates **always** based on partial knowledge
  - Uncertainty about size and factors that will impact productivity
  - Estimates need to account for the unknown (metrics can help)

Best Case Scenarios

- The same as estimating something with a 1% probability of success
- Always a bad idea to commit to something that is more likely to fail than succeed

Do You Really Want to Plan Your Project Here?



QUESTIONS?