

# 10 Keys to Successful Software Projects: An Executive Guide

© 2000-2006 Construx Software Builders, Inc.  
All Rights Reserved.

[www.construx.com](http://www.construx.com)

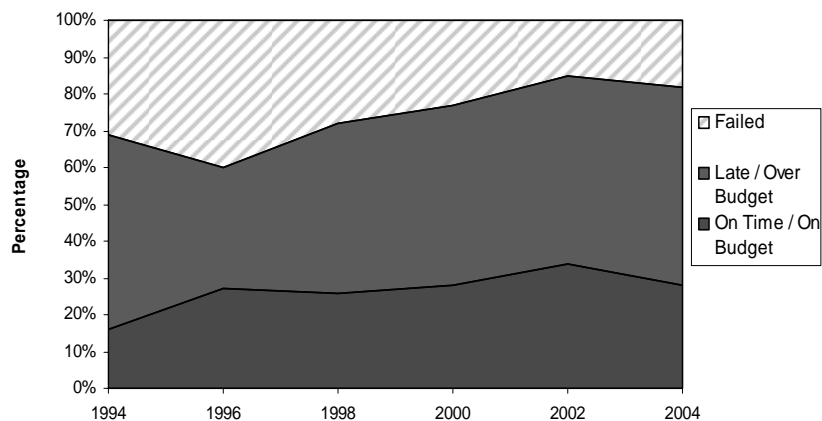
## Background

## State of the Art vs. State of the Practice

“The gap between the best software engineering practice and the average practice is very wide—perhaps wider than in any other engineering discipline.”  
– Fred Brooks

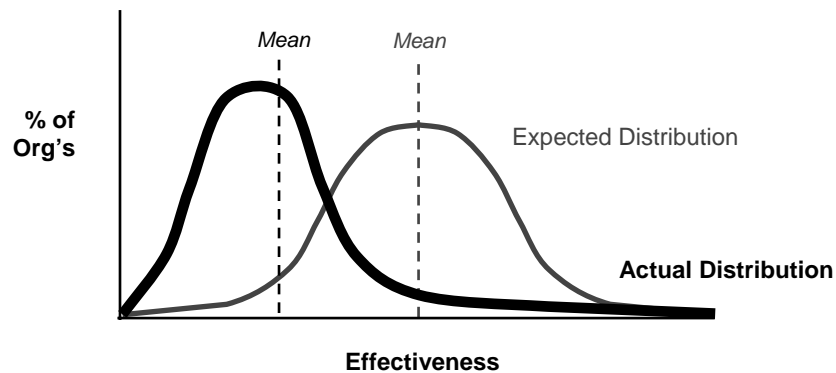
## Typical Project Outcomes

Project Outcomes 1994-2004



Source: Standish Group Chaos Report, 1994, 1996, 1998, 2000, 2002, 2004

## Average Practice is Close to the Worst Practice



## Productivity Varies Significantly

- ❖ 10:1 variations in productivity between different companies working in the same industries
- ❖ Productivity is a learned characteristic and can be changed

## **Most Common Sources of Cancellations and Overruns**

---

1. Ill-defined or changing requirements
2. Poor project planning/management
3. Uncontrolled quality problems
4. Unrealistic expectations/inaccurate estimates
5. Naive adoption of new technology

## **10 Keys to Success**

# Key #1

## Clear Vision

---

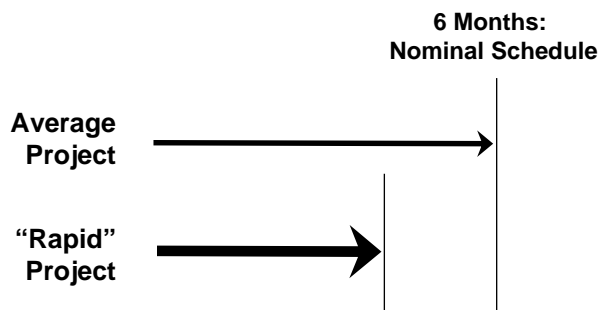
## Clear Vision

- ❖ Project teams work toward the goals you set for them
- ❖ Too many goals = no goals
- ❖ Good vision statement describes *what to leave out*—prioritizes
- ❖ Product vision affects achievement of business goals

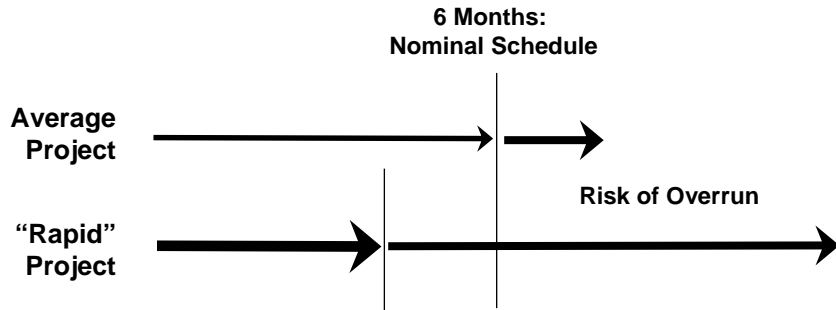
## Example:

- ❖ What kind of Rapid Development do you want?
  - ◆ Speed oriented
  - ◆ Schedule-risk oriented
  - ◆ Visibility oriented
- ❖ Without clear vision you can end up with a project outcome completely counter to your goals

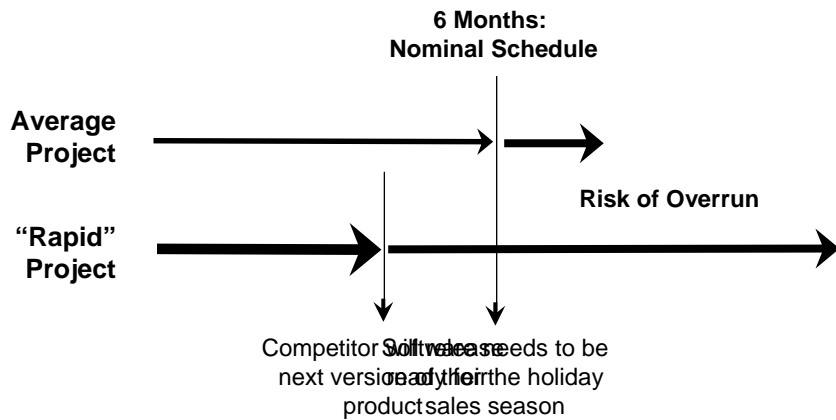
## Speed-Oriented Practices-- Better Best Case ...



## ...Worse Worst Case



## Sometimes it's Justified (sometimes not)



## Key #2

### **Stable, Complete, Written Requirements**

## **Requirements**

- ❖ Requirements change is the most common software project risk
- ❖ Comprehensive, 100% stable requirements are usually not possible, but...
- ❖ Most requirements changes arise from requirements that were incompletely defined in the first place, not “changing markets” or other similar reasons

## Techniques for Defining Stable Requirements

---

- ❖ Requirements workshop
- ❖ User interface prototyping
- ❖ User interview
- ❖ Use cases
- ❖ User manual as spec
- ❖ Usability studies
- ❖ Incremental delivery
- ❖ Requirements reviews/inspections

### **Key #3**

## **Detailed User Interface Prototypes**

## User Interface Prototypes

---

- ❖ Addresses the most common project risk—changing requirements
- ❖ Involves users with a “hot” medium
- ❖ Correlated with lower costs, shorter schedules, and higher user satisfaction
- ❖ Significant skill required to develop prototypes successfully

## Key #4

## Effective Project Management

## Project Management

- ❖ Poor planning/management is the second most common project risk
- ❖ Project planning/management is a high leverage area
- ❖ Some people don't appreciate the leverage of project management—they've never seen good project management!

## Project Manager Responsibilities

- ❖ Where do most project managers come from?
- ❖ What are they trained to do?
- ❖ Good software management require significant software-specific expertise
  - ◆ Scope estimation
  - ◆ Cost, effort & schedule computation
  - ◆ Lifecycle selection
  - ◆ QA planning
  - ◆ Technical staffing
  - ◆ Project tracking
  - ◆ Risk management
  - ◆ Data collection

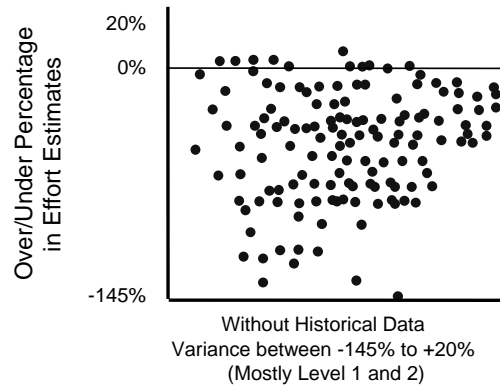
## Key #5

### Accurate Estimates

## Need for Accurate Estimates

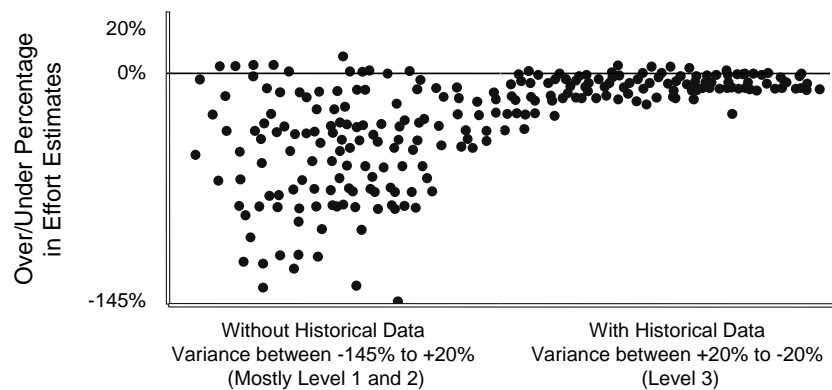
- ❖ Unrealistic/unjustified expectations are a major cause of project problems
- ❖ State of the art is dramatically better than the state of the practice
- ❖ For example, the average project overruns its planned schedule by more than 100% (and many projects are much worse)

# Typical Estimation Effectiveness



From 120 Projects in Boeing Information Systems

# Improved Estimation



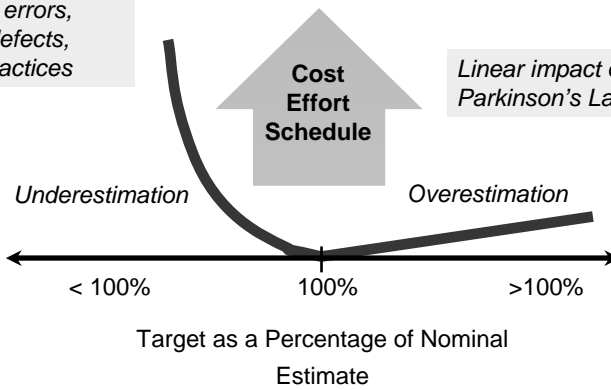
From 120 Projects in Boeing Information Systems

# Improved Estimation



# Effect of Estimation Accuracy

*Non-linear impact due to planning errors, upstream defects, high-risk practices*



## Accurate Estimation

---

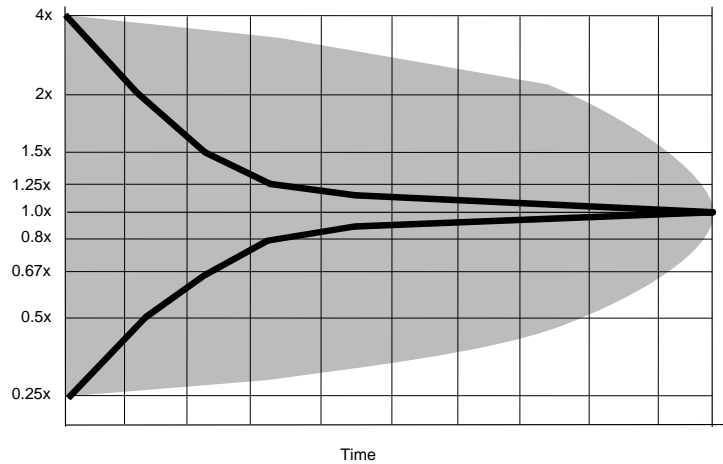
- ❖ Estimation is a specialized technical skill
- ❖ Treat estimation as a mini-project
- ❖ More on this topic momentarily ...

## Key #6

### Two-Phase Budgeting

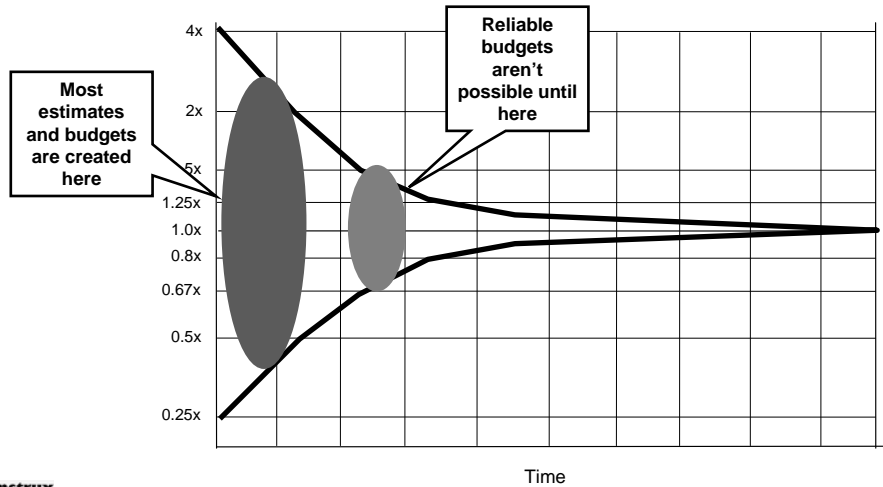
# Cone of Uncertainty and Estimate Refinement

Project Scope  
(effort, cost, or features)

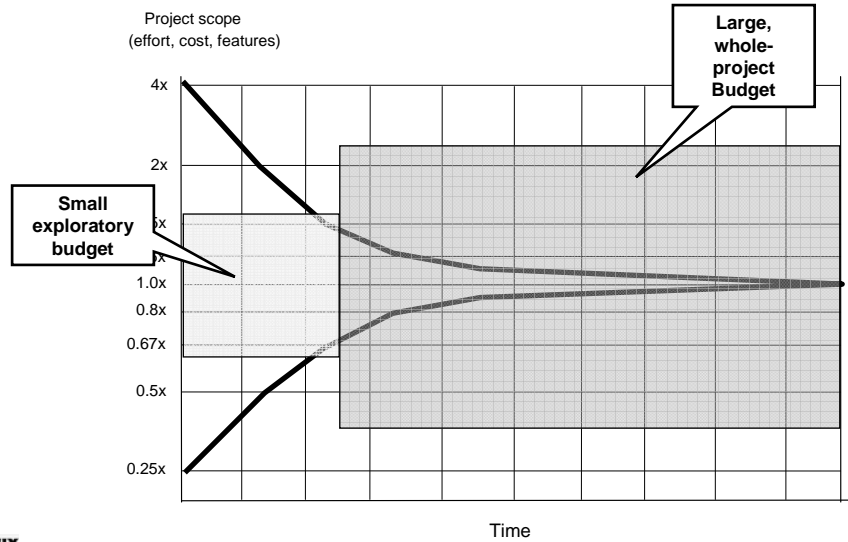


# Good and Bad Budgeting

Project scope  
(effort, cost, features)



## Two-Phase Budgeting



## Benefits of 2-Phase Budgeting

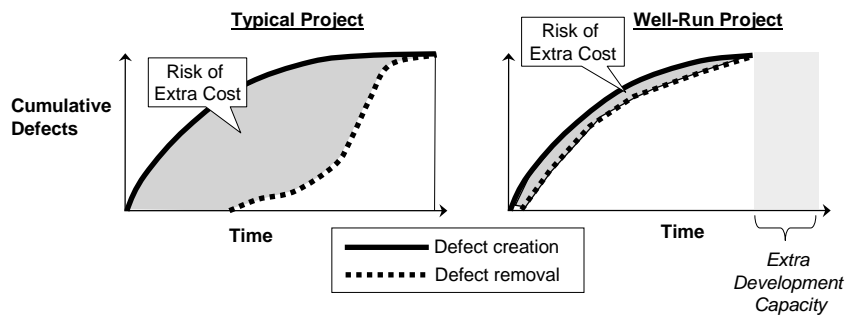
- ❖ Delays commitment until time when a commitment can be meaningful
- ❖ Forces activities that should occur upstream actually to occur upstream
  - ◆ Requirements, technical planning, quality planning, etc.
- ❖ Helps set realistic expectations for all project stakeholders
- ❖ Improves coordination with non-software groups
- ❖ Improves execution by putting plans on more informed basis

# Key #7

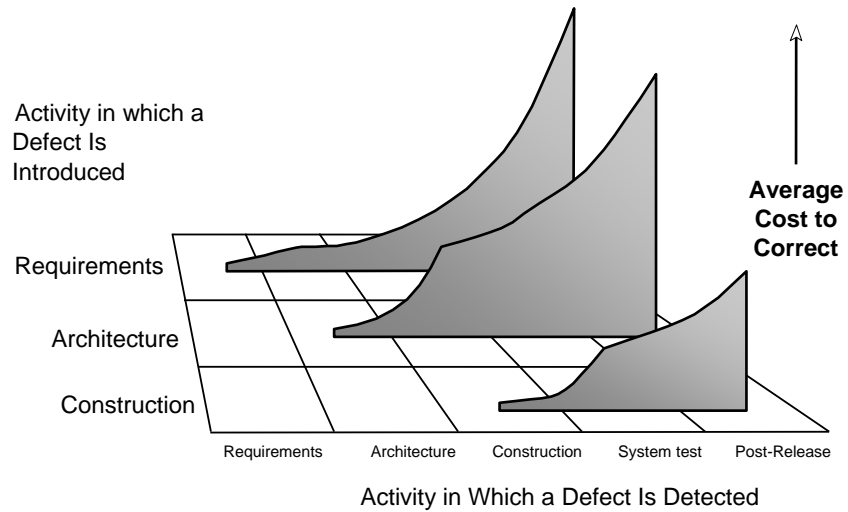
## A Focus on Quality

### General Principle

- ❖ Defect creation is a function of effort
- ❖ Defect detection is a function of QA activities
- ❖ Goal is to minimize gap between defect insertion and defect detection/correction



# Defect Cost Increase (DCI)

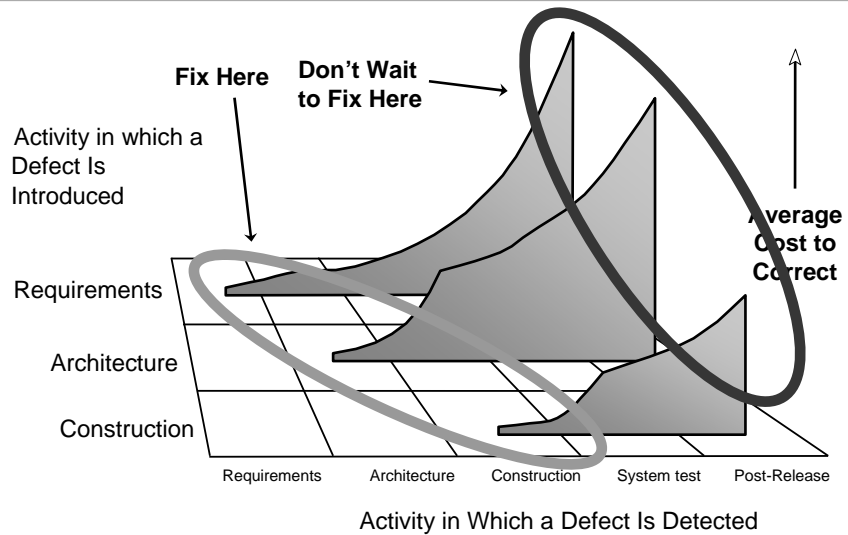


Construx

"Software Development Best Practices"

37

# One Solution: Fix Defects Earlier!

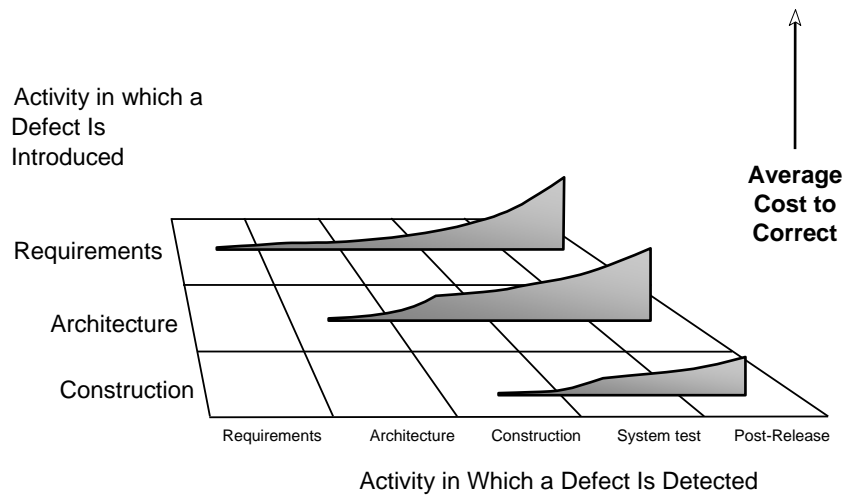


Construx

"Software Development Best Practices"

38

## Another Solution: Reduce Defect Cost Increase!

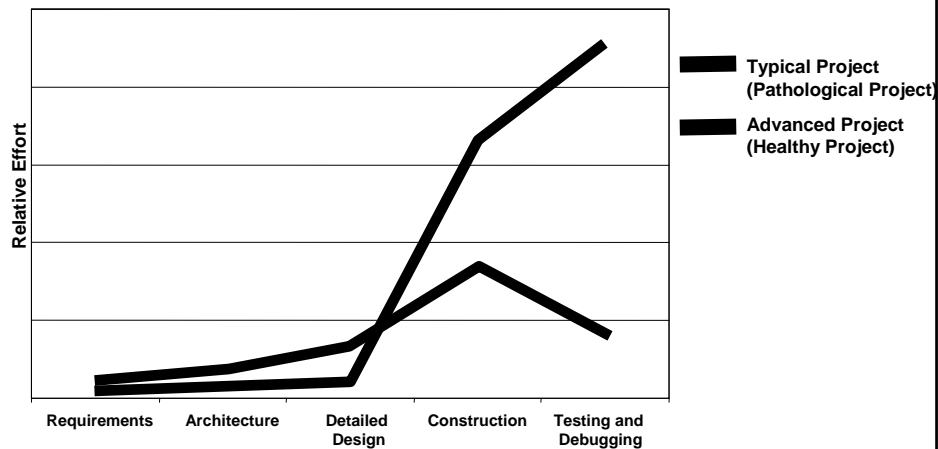


Construx

"Software Development Best Practices"

39

## Where Costs Come From: Lifecycle Cost Profile



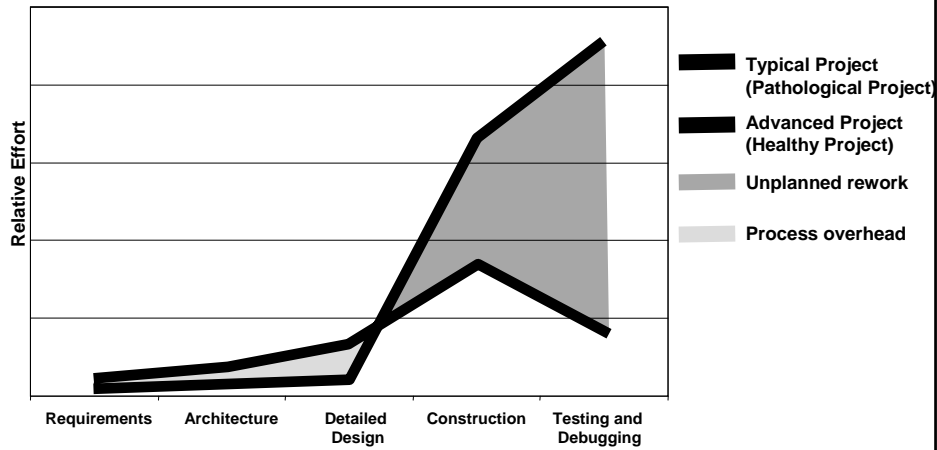
Construx

"Software Development Best Practices"

40

Where Costs Come From:

## Lifecycle Cost Profile (cont.)



Construx

"Software Development Best Practices"

41

## Why Focus on Quality?

- ❖ For most projects, unplanned defect correction work is the largest cost driver (40-80% of total)
- ❖ Can focus on quality for sake of economics (as above)
- ❖ Can focus on quality for sake of quality (not needed nearly as often)
- ❖ Quality must be planned into the project; it can't just be tacked onto the end

Construx

"Software Development Best Practices"

42

## Key #8

### Technology Expertise

## Technology Expertise

- ❖ Many projects suffer because of poor adoption of new technology
- ❖ “New technology” = high risk
- ❖ Golf analogy
  - ◆ Technology as golf clubs
  - ◆ Best practices as technique
- ❖ Expertise in technology matters; software engineering technique matters much more

## Key #9

### Active Risk Management

## Software is Risky Business

- ❖ KPMG Study:
  - ◆ 55% of runaway projects did no risk management
  - ◆ 38% did some, but half of those didn't use their risk findings after the project was underway
  - ◆ 7% didn't know whether they did risk management
- ❖ Total: About 80% of runaway projects did no meaningful risk management

## Role of Risk Management

- ❖ About as many projects fail as are delivered on time
- ❖ More than 50% of projects show their problems during initial development
- ❖ About 25% show their problems during initial planning
- ❖ Active risk management keeps small problems from turning into big, project-killing problems

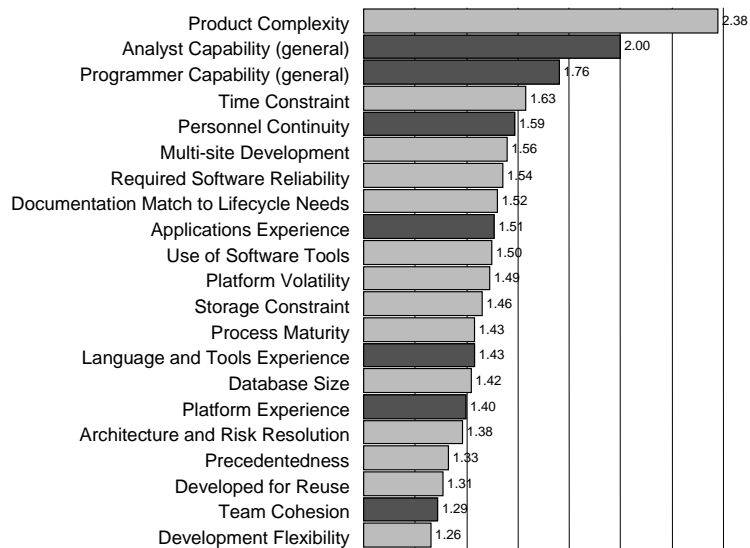
## Relationship to Business Risk Taking

- ❖ Perception is that high-energy companies take risks
- ❖ Reality is that most companies are beset by risks from all sides—they aren't choosing which risks they take
- ❖ Key to success: Manage non-strategic risks so that you can take strategic risks

# Key #10

**Remember, Software Is  
Created By Humans**

## **Cocomo II's View of Software Project Influences**



## Importance of Human Influences

- ❖ Human Influences make a 14x difference in total project effort & cost, according to Cocomo II
- ❖ Capability factors alone make a 3.5x difference
- ❖ Experience factors alone make a 3.0x difference
- ❖ Consensus of studies is that similarly-experienced developers vary by about 20x in productivity and quality of work

## Where do Variations Exist?

Researchers have found 20:1 variations in:

- ❖ Coding speed
- ❖ Debugging speed
- ❖ Defect-finding speed
- ❖ Percentage of defects found
- ❖ Bad-fix injection rate
- ❖ Design quality
- ❖ Amount of code generated from a design
- ❖ Teamwork
- ❖ Etc.

## Some Implications

---

- ❖ Success of Google, Amazon, Microsoft, etc.
- ❖ Matching workers and workstyles
- ❖ Value of retention programs
- ❖ Importance of staff development

## Conclusion

# Project Success

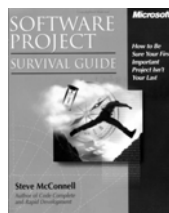
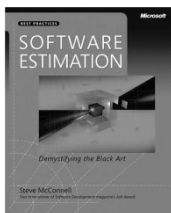
Success = Planning \* Execution

Assign values between 0%-100% for *Planning* and *Execution*. Multiply to determine your chance of success.

# Contact Info

Steve McConnell

Construx Software  
10900 NE 8<sup>th</sup> Street, Suite 1350  
Bellevue, WA 98005  
(425) 636-0100  
stevemcc@construx.com



❖ **Training**  
❖ **Consulting**  
❖ **Tools**

**sales@construx.com**  
**www.construx.com**  
**+1 (425) 636-0100**