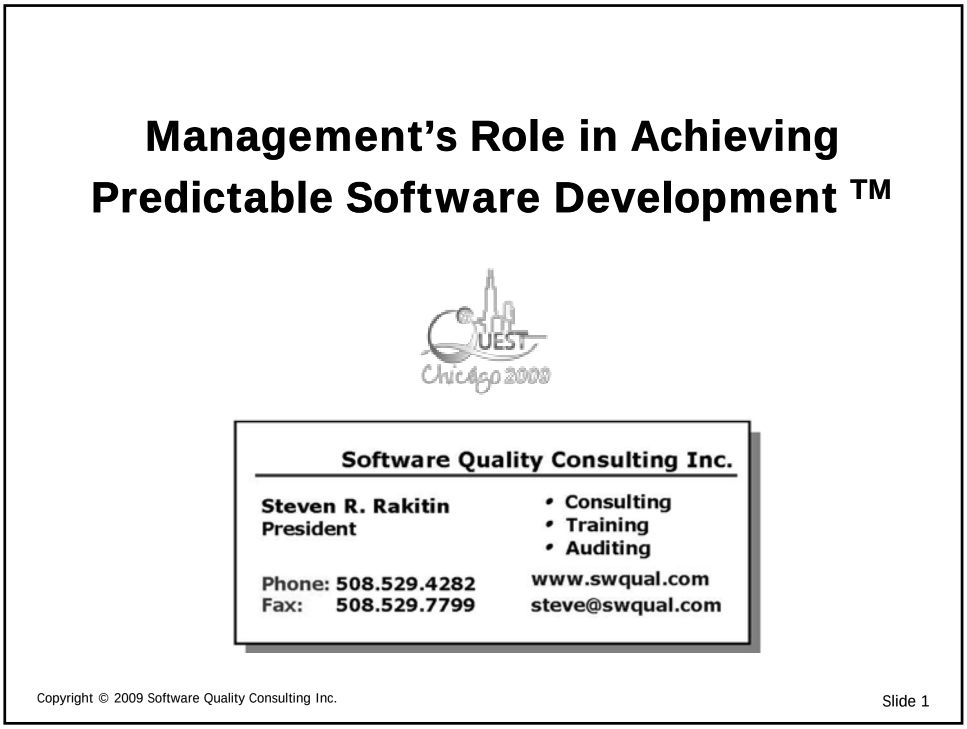


Management’s Role in Achieving Predictable Software Development ™

QUEST Chicago 2009

Software Quality Consulting Inc.

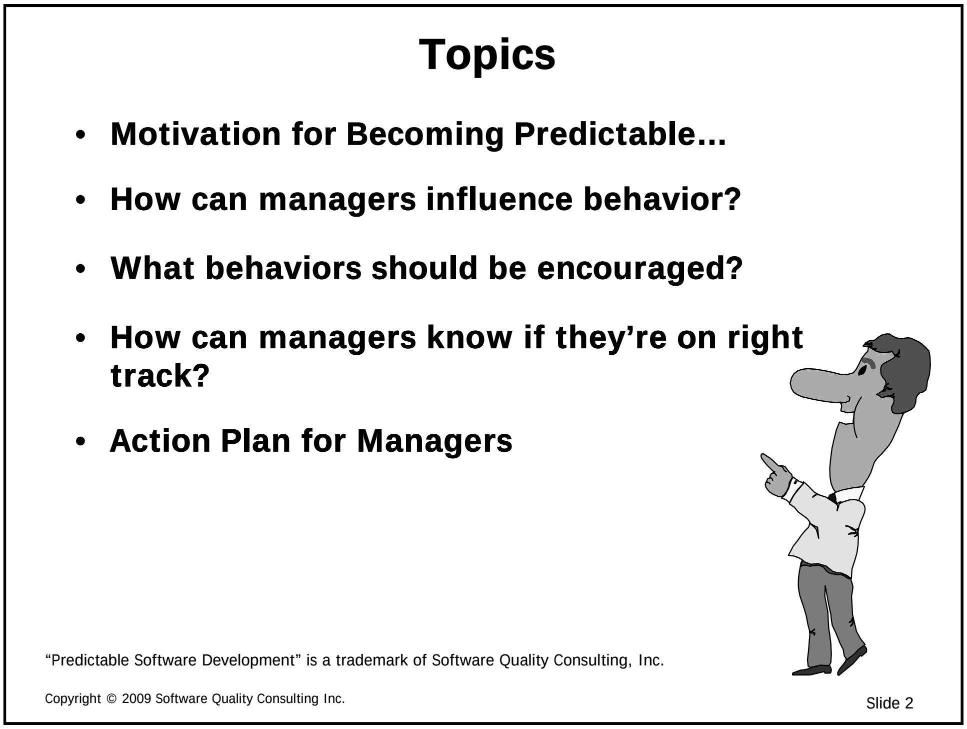
Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
steve@swqual.com

Slide 1

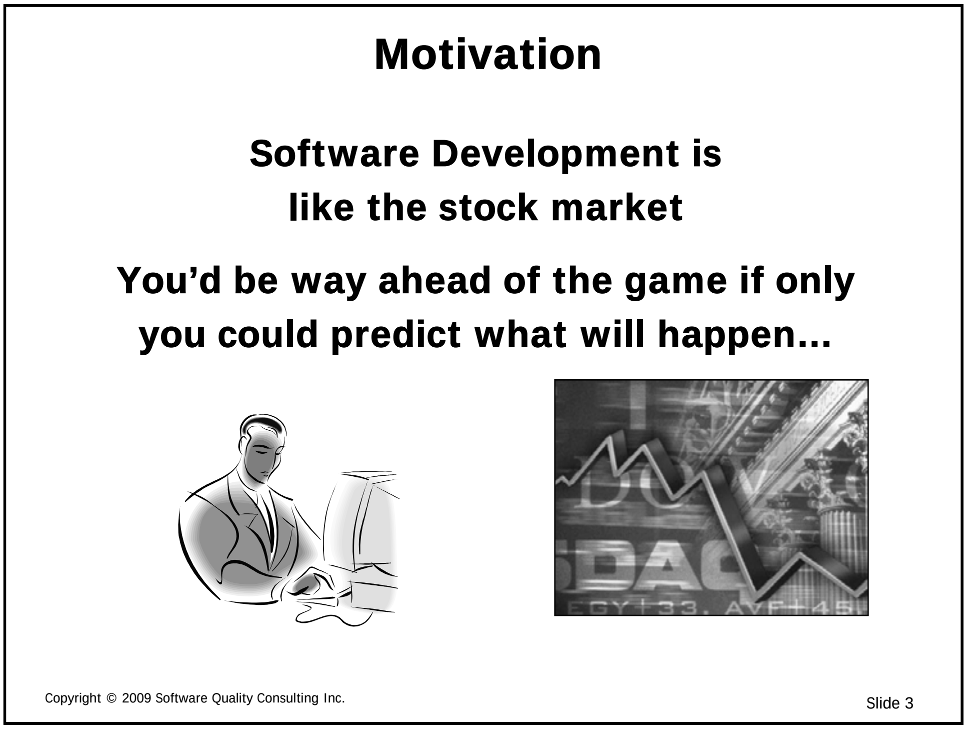


Topics

- Motivation for Becoming Predictable…
- How can managers influence behavior?
- What behaviors should be encouraged?
- How can managers know if they’re on right track?
- Action Plan for Managers

“Predictable Software Development” is a trademark of Software Quality Consulting, Inc.

Slide 2

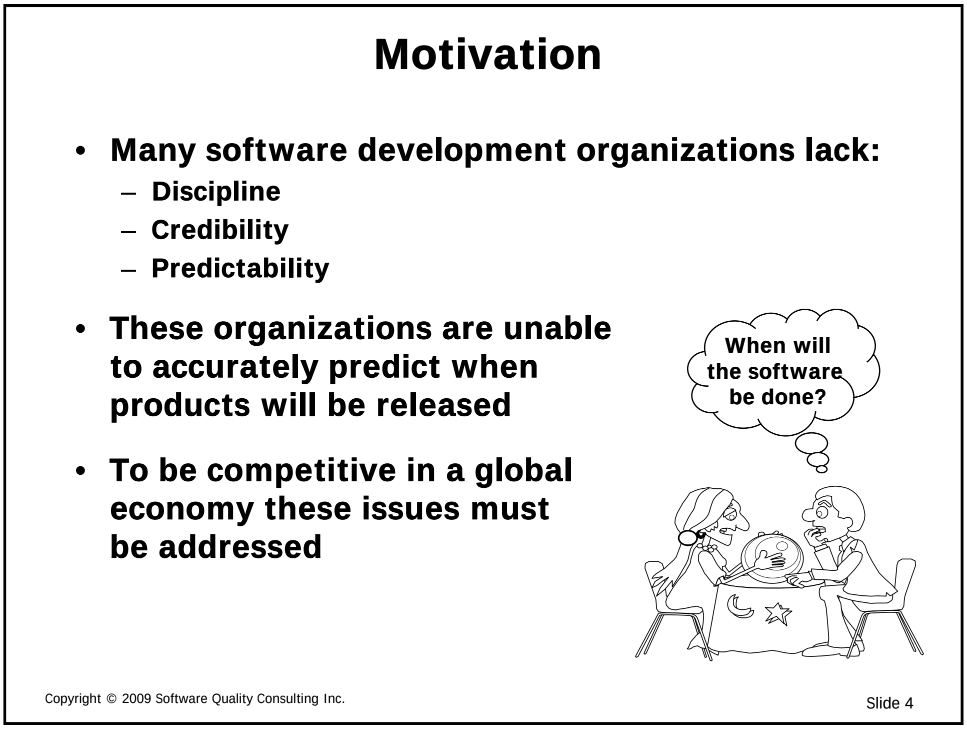


Motivation

Software Development is like the stock market

You’d be way ahead of the game if only you could predict what will happen...

Slide 3

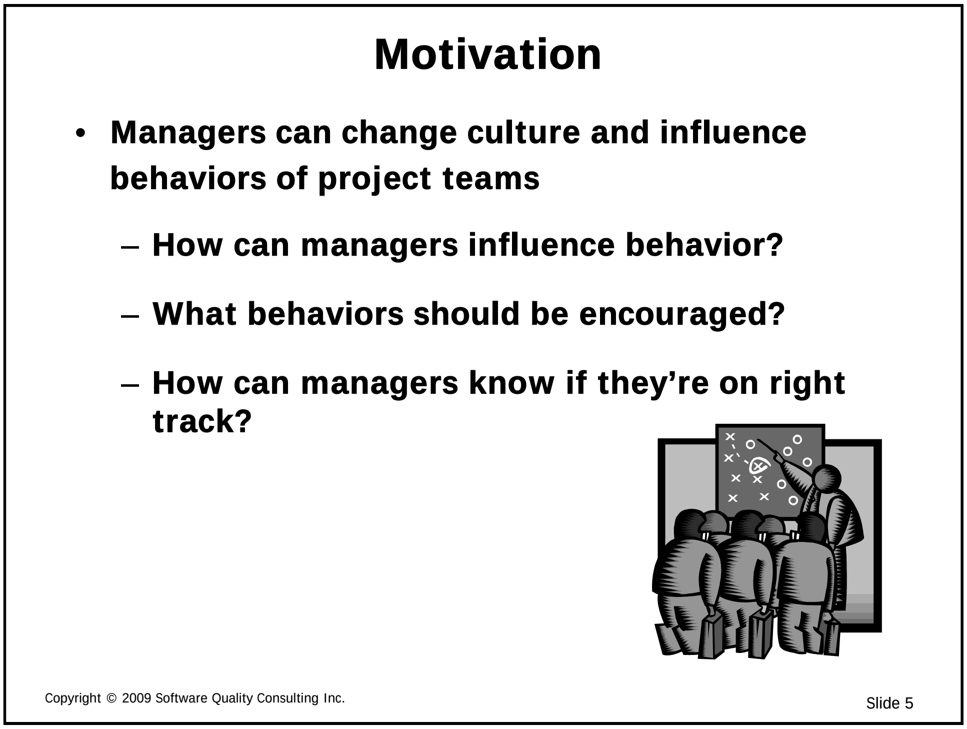


Motivation

- **Many software development organizations lack:**
  - **Discipline**
  - **Credibility**
  - **Predictability**
- **These organizations are unable to accurately predict when products will be released**
- **To be competitive in a global economy these issues must be addressed**

When will the software be done?

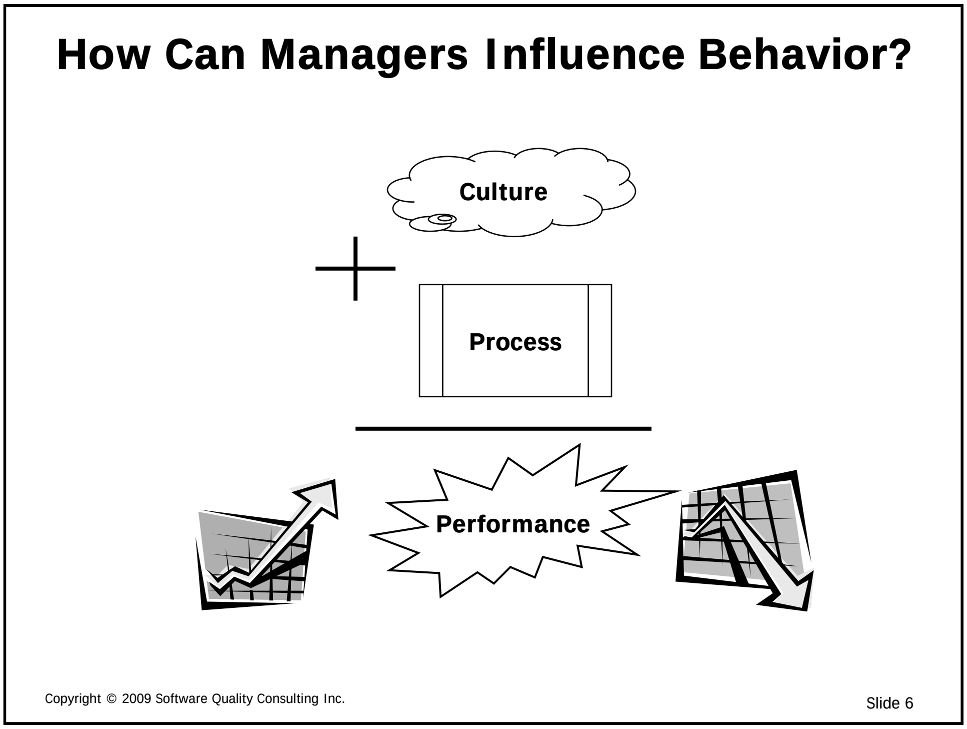
Slide 4



Motivation

- Managers can change culture and influence behaviors of project teams
  - How can managers influence behavior?
  - What behaviors should be encouraged?
  - How can managers know if they’re on right track?

Slide 5



How Can Managers Influence Behavior?

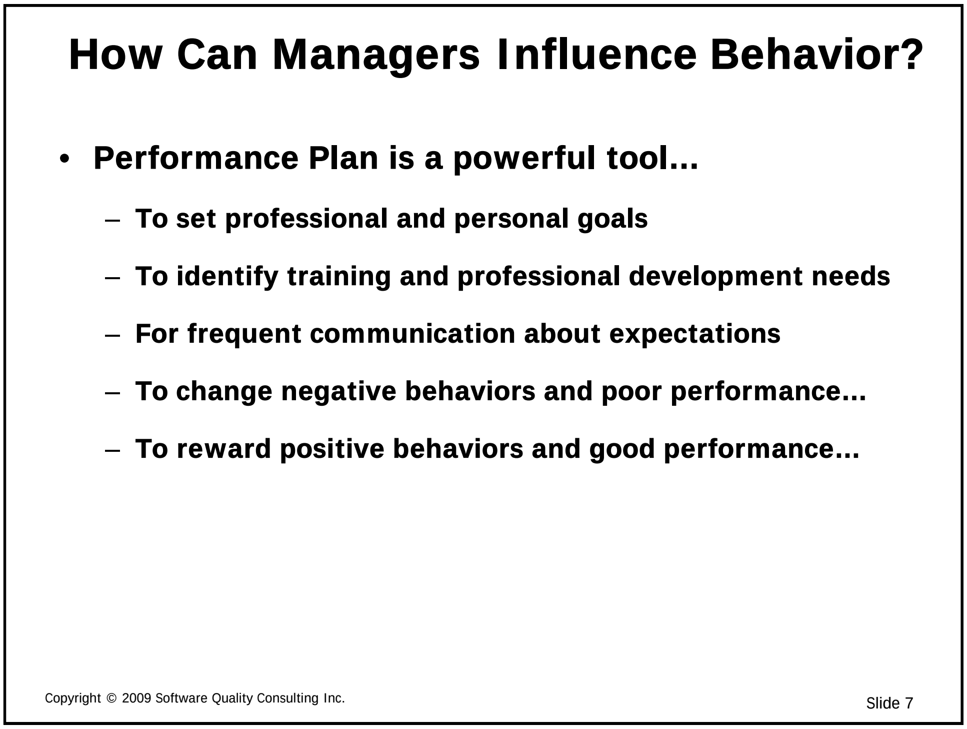
Culture

+

Process

Performance

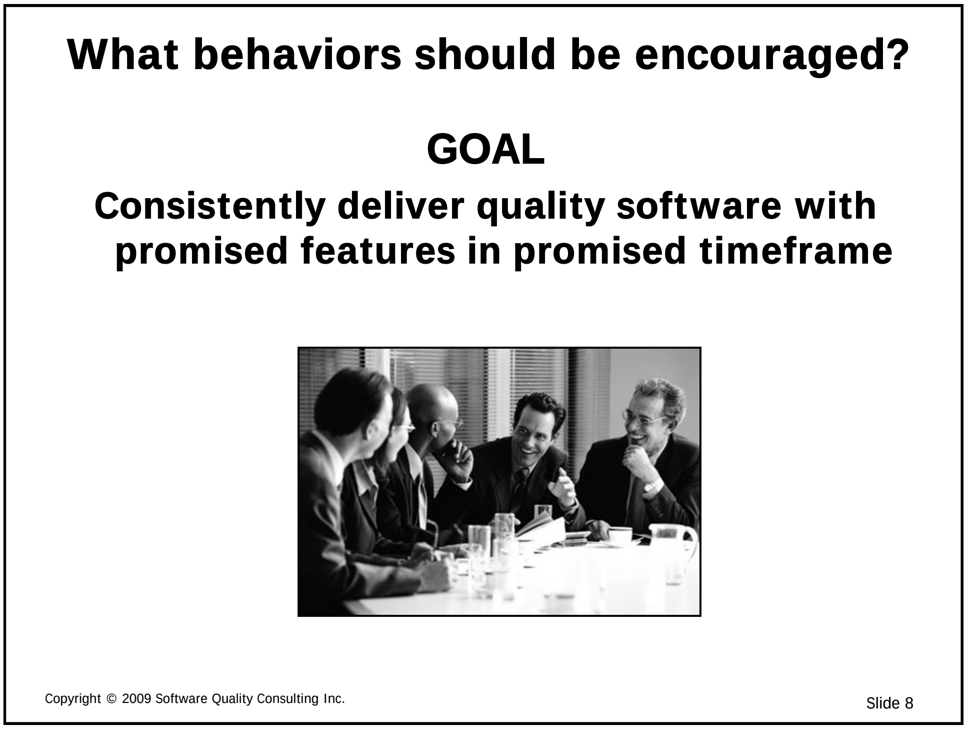
Slide 6



How Can Managers Influence Behavior?

- **Performance Plan is a powerful tool…**
  - **To set professional and personal goals**
  - **To identify training and professional development needs**
  - **For frequent communication about expectations**
  - **To change negative behaviors and poor performance…**
  - **To reward positive behaviors and good performance…**

Slide 7

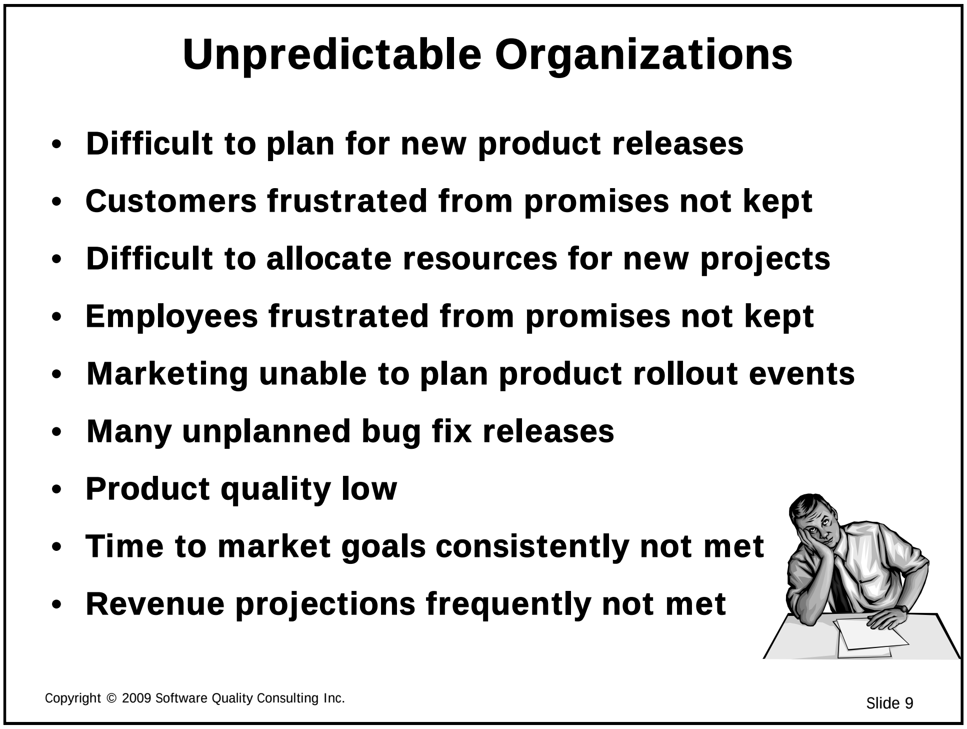


What behaviors should be encouraged?

GOAL

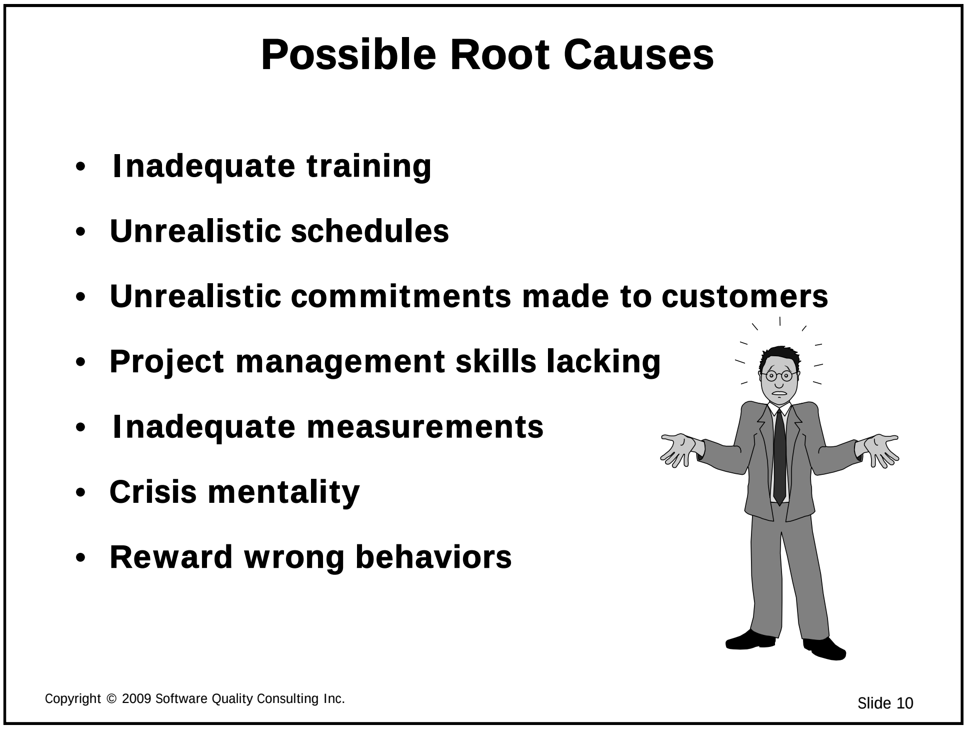
Consistently deliver quality software with promised features in promised timeframe

Slide 8



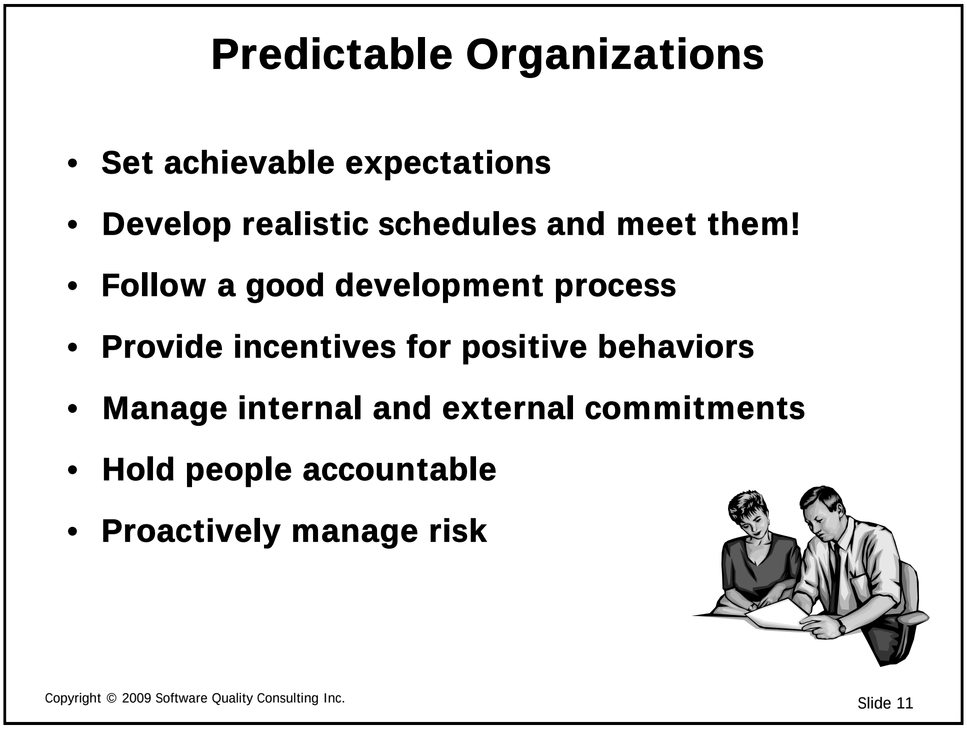
Unpredictable Organizations

- Difficult to plan for new product releases
- Customers frustrated from promises not kept
- Difficult to allocate resources for new projects
- Employees frustrated from promises not kept
- Marketing unable to plan product rollout events
- Many unplanned bug fix releases
- Product quality low
- Time to market goals consistently not met
- Revenue projections frequently not met



Possible Root Causes

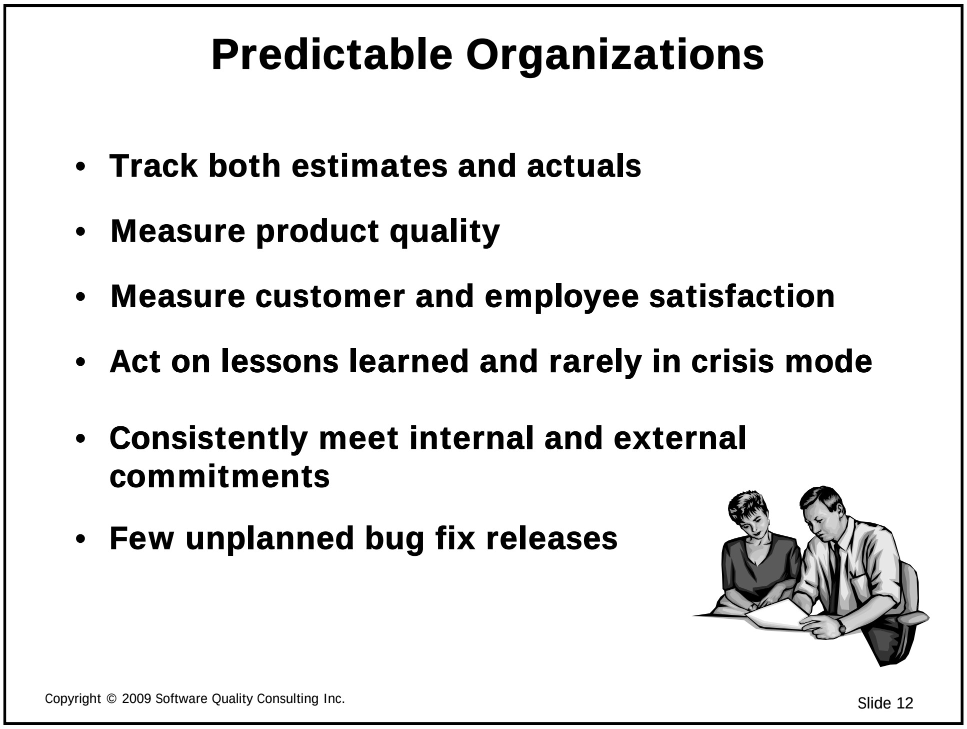
- Inadequate training
- Unrealistic schedules
- Unrealistic commitments made to customers
- Project management skills lacking
- Inadequate measurements
- Crisis mentality
- Reward wrong behaviors



Predictable Organizations

- Set achievable expectations
- Develop realistic schedules and meet them!
- Follow a good development process
- Provide incentives for positive behaviors
- Manage internal and external commitments
- Hold people accountable
- Proactively manage risk

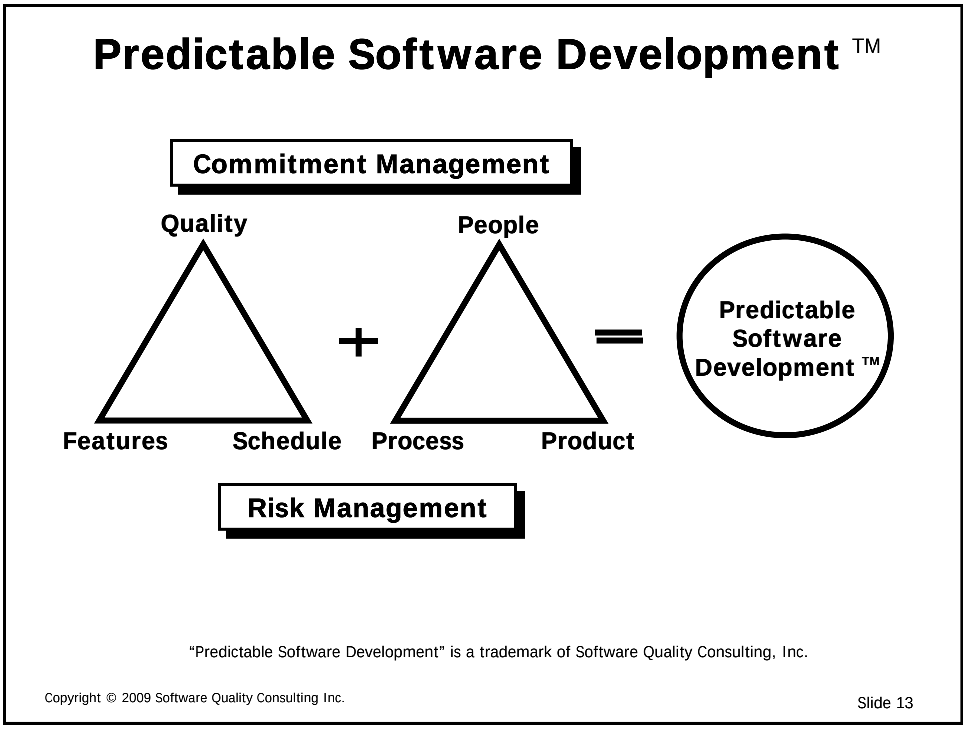
Slide 11



Predictable Organizations

- Track both estimates and actuals
- Measure product quality
- Measure customer and employee satisfaction
- Act on lessons learned and rarely in crisis mode
- Consistently meet internal and external commitments
- Few unplanned bug fix releases

Slide 12



Predictable Software Development ™

Commitment Management

Quality

Features

Schedule

+

People

Process

Product

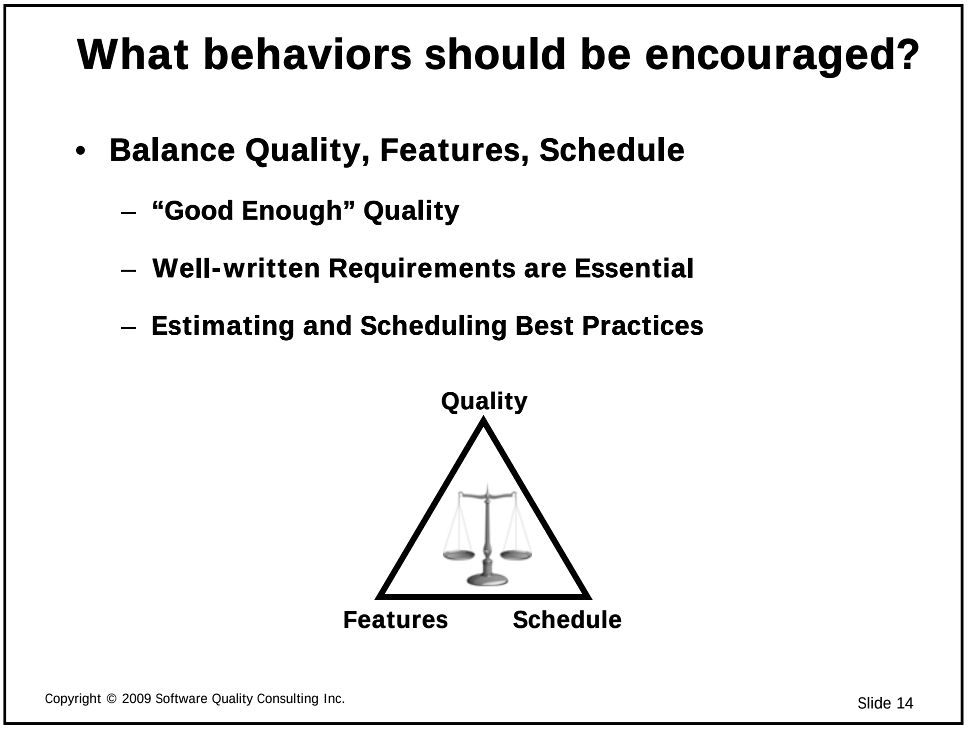
=

Predictable Software Development ™

Risk Management

"Predictable Software Development" is a trademark of Software Quality Consulting, Inc.

Slide 13



What behaviors should be encouraged?

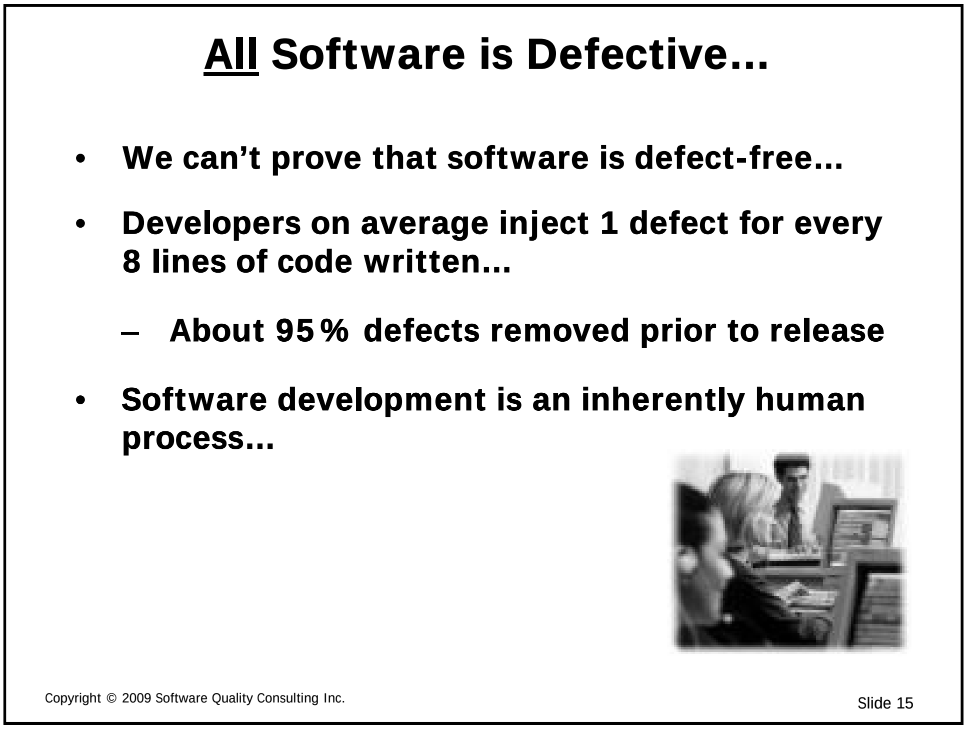
- **Balance Quality, Features, Schedule**
  - **"Good Enough" Quality**
  - **Well-written Requirements are Essential**
  - **Estimating and Scheduling Best Practices**

Quality

Features

Schedule

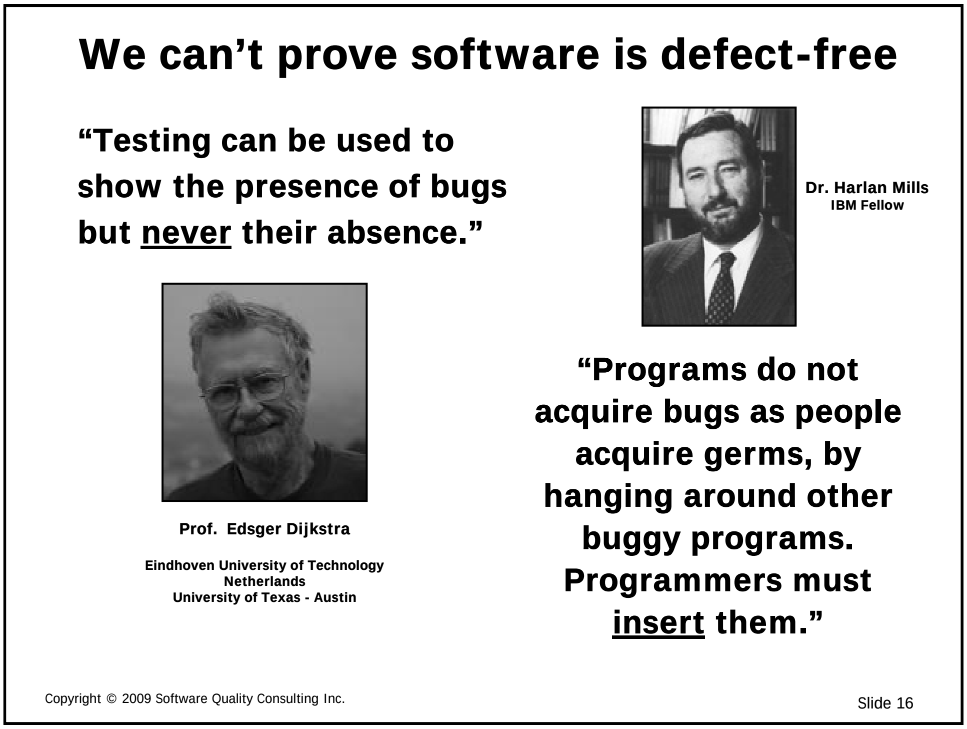
Slide 14



All Software is Defective…

- We can't prove that software is defect-free…
- Developers on average inject 1 defect for every 8 lines of code written…
  - About 95% defects removed prior to release
- Software development is an inherently human process…

Slide 15



We can't prove software is defect-free

"Testing can be used to show the presence of bugs but never their absence."

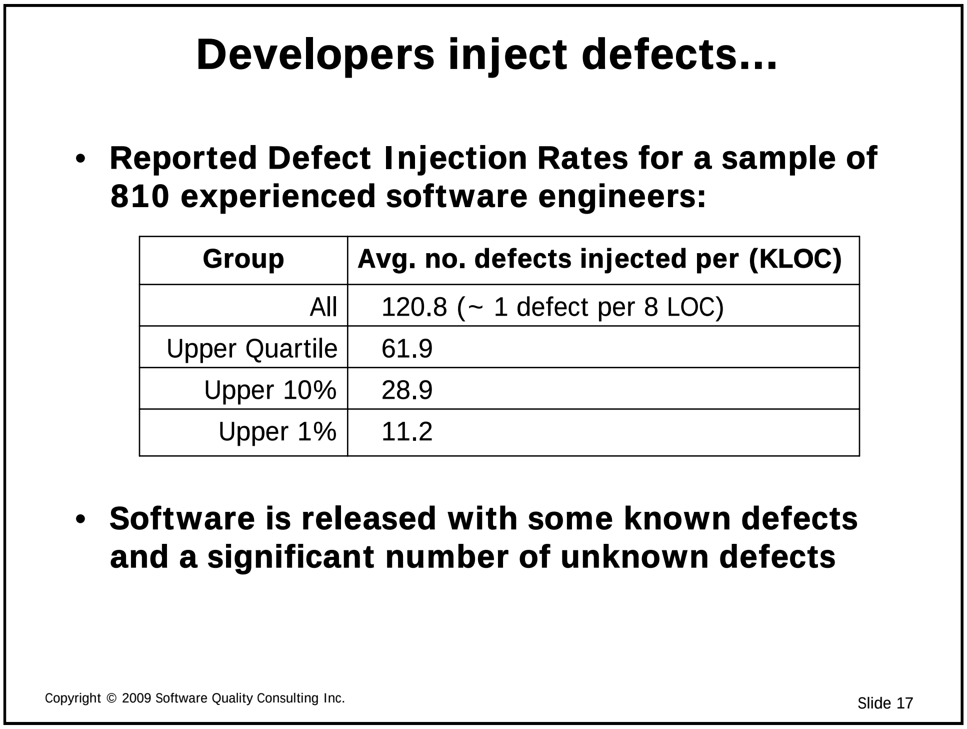
Prof. Edsger Dijkstra

Eindhoven University of Technology
Netherlands
University of Texas - Austin

Dr. Harlan Mills
IBM Fellow

"Programs do not acquire bugs as people acquire germs, by hanging around other buggy programs. Programmers must insert them."

Slide 16



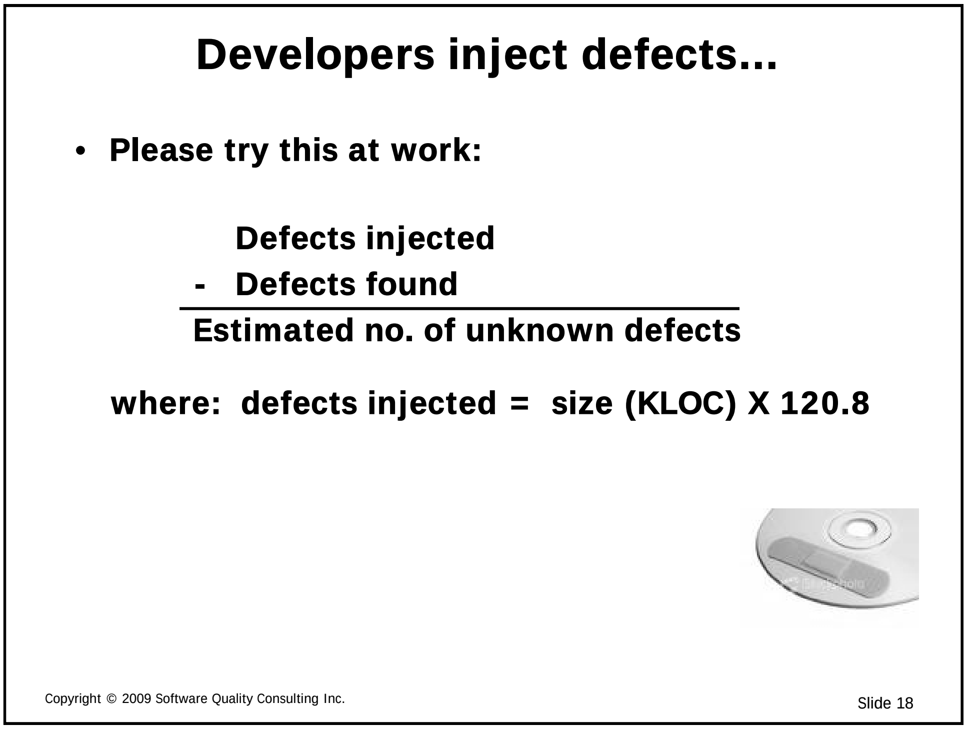
Developers inject defects...

- Reported Defect Injection Rates for a sample of 810 experienced software engineers:

Group	Avg. no. defects injected per (KLOC)
All	120.8 (~ 1 defect per 8 LOC)
Upper Quartile	61.9
Upper 10%	28.9
Upper 1%	11.2

- Software is released with some known defects and a significant number of unknown defects

Slide 17



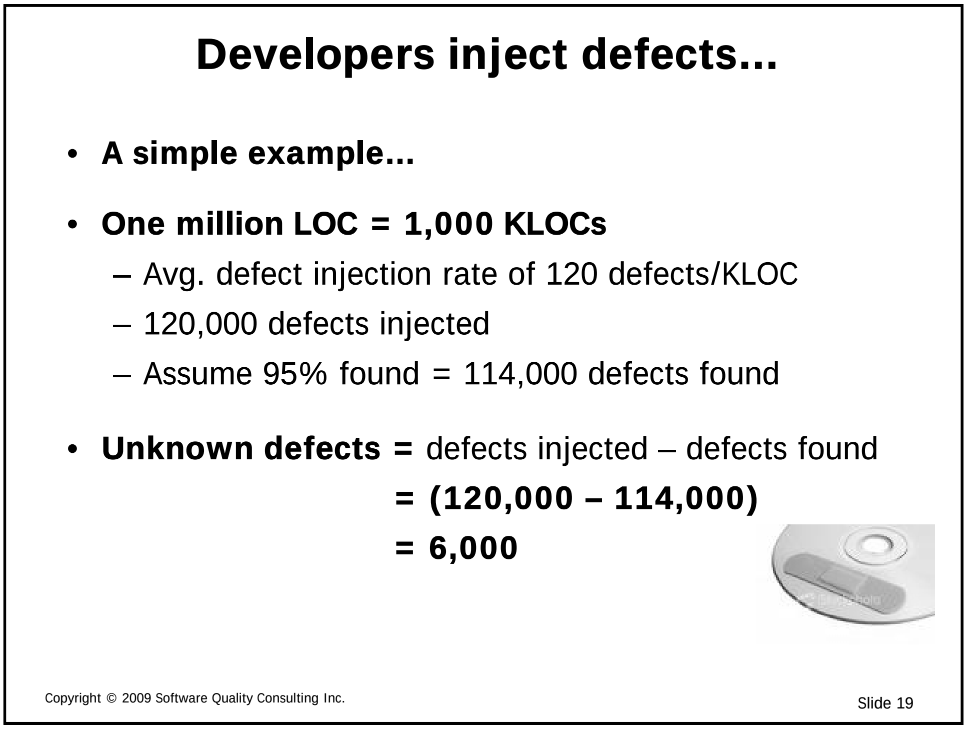
Developers inject defects...

- Please try this at work:

$$\begin{array}{rl} & \text{Defects injected} \\ - & \text{Defects found} \\ \hline & \text{Estimated no. of unknown defects} \end{array}$$

where: defects injected = size (KLOC) X 120.8

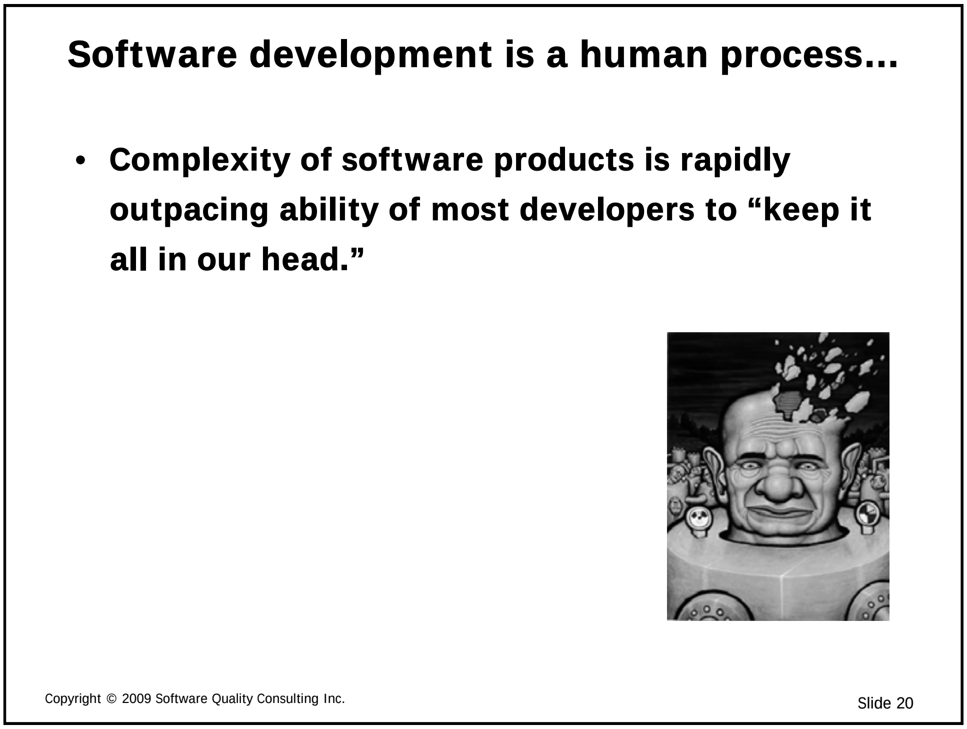
Slide 18



Developers inject defects...

- **A simple example...**
- **One million LOC = 1,000 KLOCs**
  – Avg. defect injection rate of 120 defects/KLOC
  – 120,000 defects injected
  – Assume 95% found = 114,000 defects found
- **Unknown defects =** defects injected – defects found
  = (120,000 – 114,000)
  = 6,000

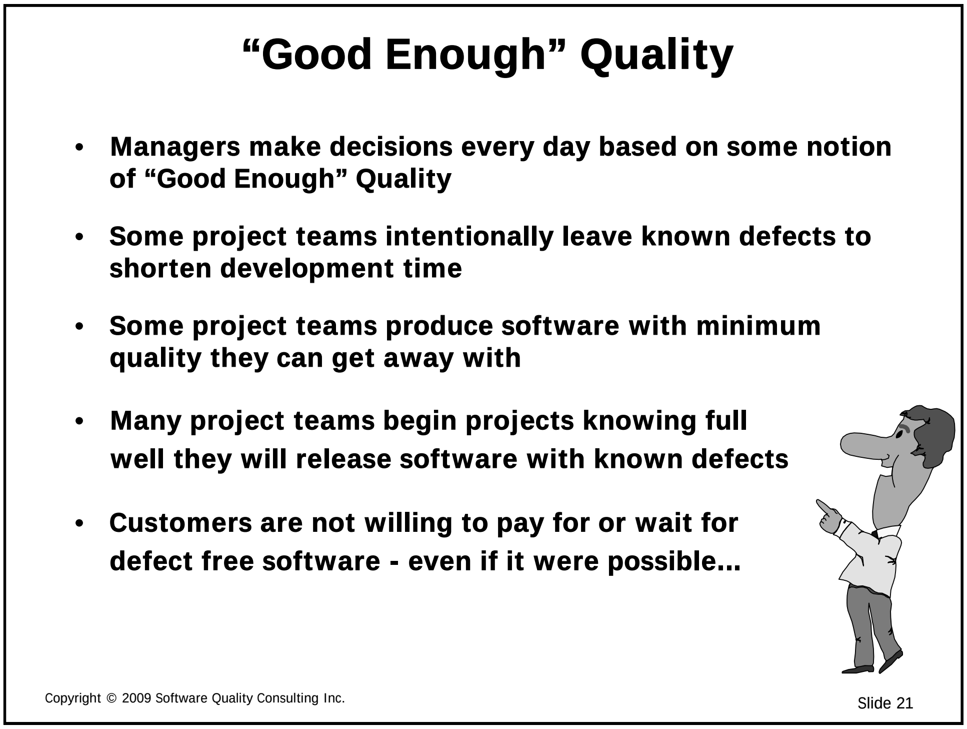
Slide 19



Software development is a human process...

- **Complexity of software products is rapidly outpacing ability of most developers to “keep it all in our head.”**

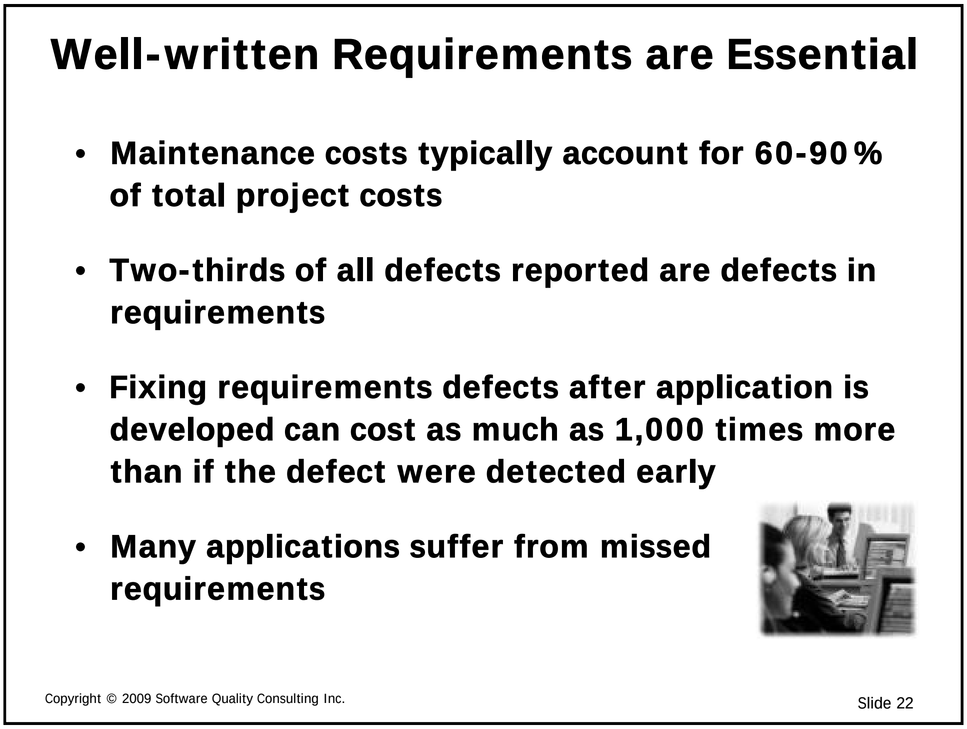
Slide 20



“Good Enough” Quality

- Managers make decisions every day based on some notion of “Good Enough” Quality
- Some project teams intentionally leave known defects to shorten development time
- Some project teams produce software with minimum quality they can get away with
- Many project teams begin projects knowing full well they will release software with known defects
- Customers are not willing to pay for or wait for defect free software - even if it were possible...

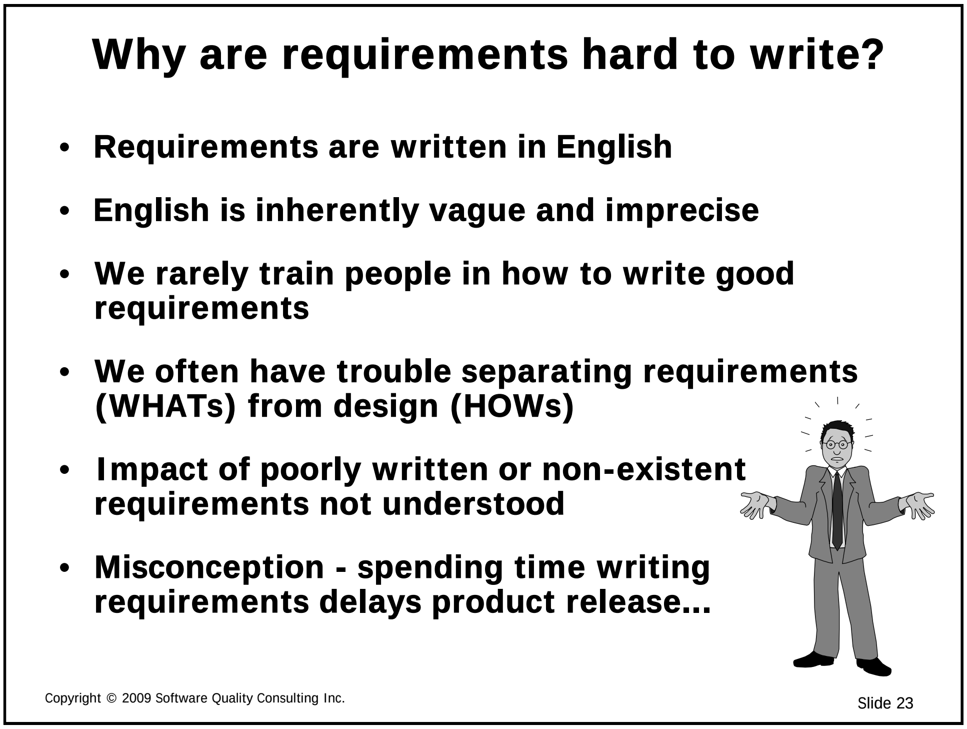
Slide 21



Well-written Requirements are Essential

- Maintenance costs typically account for 60-90% of total project costs
- Two-thirds of all defects reported are defects in requirements
- Fixing requirements defects after application is developed can cost as much as 1,000 times more than if the defect were detected early
- Many applications suffer from missed requirements

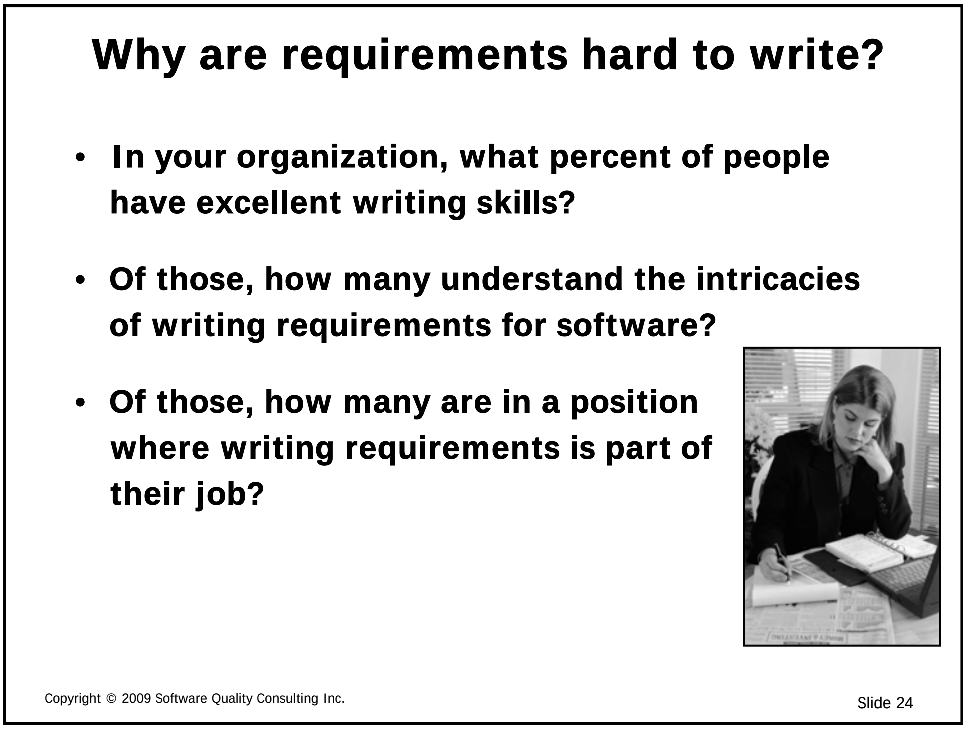
Slide 22



Why are requirements hard to write?

- Requirements are written in English
- English is inherently vague and imprecise
- We rarely train people in how to write good requirements
- We often have trouble separating requirements (WHATs) from design (HOWs)
- Impact of poorly written or non-existent requirements not understood
- Misconception - spending time writing requirements delays product release...

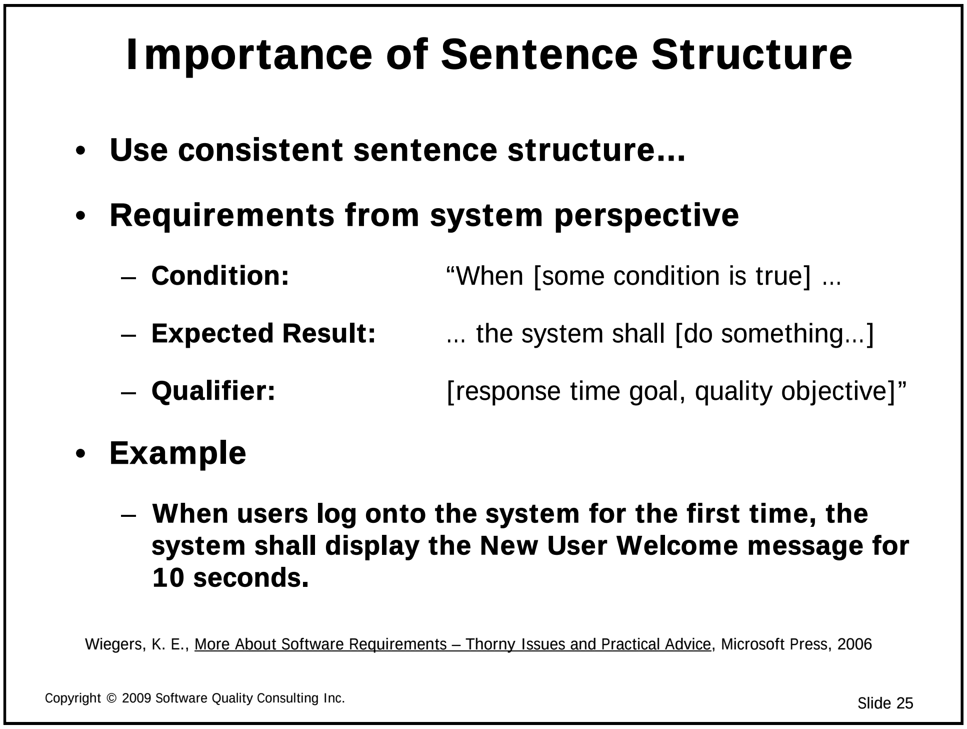
Slide 23



Why are requirements hard to write?

- In your organization, what percent of people have excellent writing skills?
- Of those, how many understand the intricacies of writing requirements for software?
- Of those, how many are in a position where writing requirements is part of their job?

Slide 24

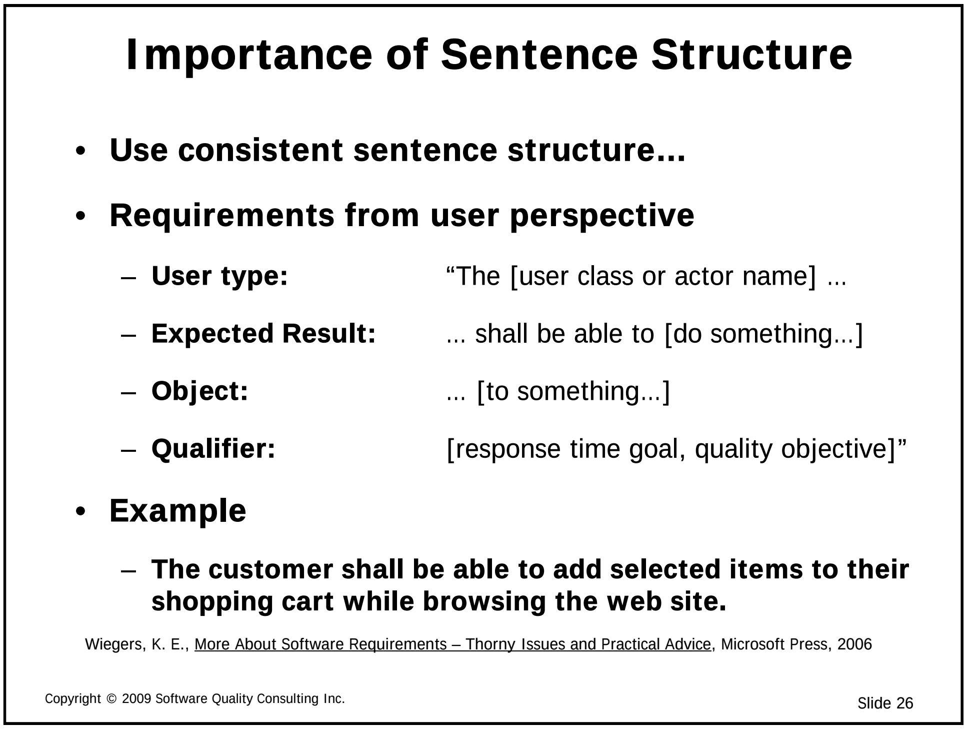


Importance of Sentence Structure

- **Use consistent sentence structure…**
- **Requirements from system perspective**
  - **Condition:** “When [some condition is true] …
  - **Expected Result:** … the system shall [do something...]
  - **Qualifier:** [response time goal, quality objective]”
- **Example**
  - **When users log onto the system for the first time, the system shall display the New User Welcome message for 10 seconds.**

Wiegers, K. E., More About Software Requirements – Thorny Issues and Practical Advice, Microsoft Press, 2006

Slide 25

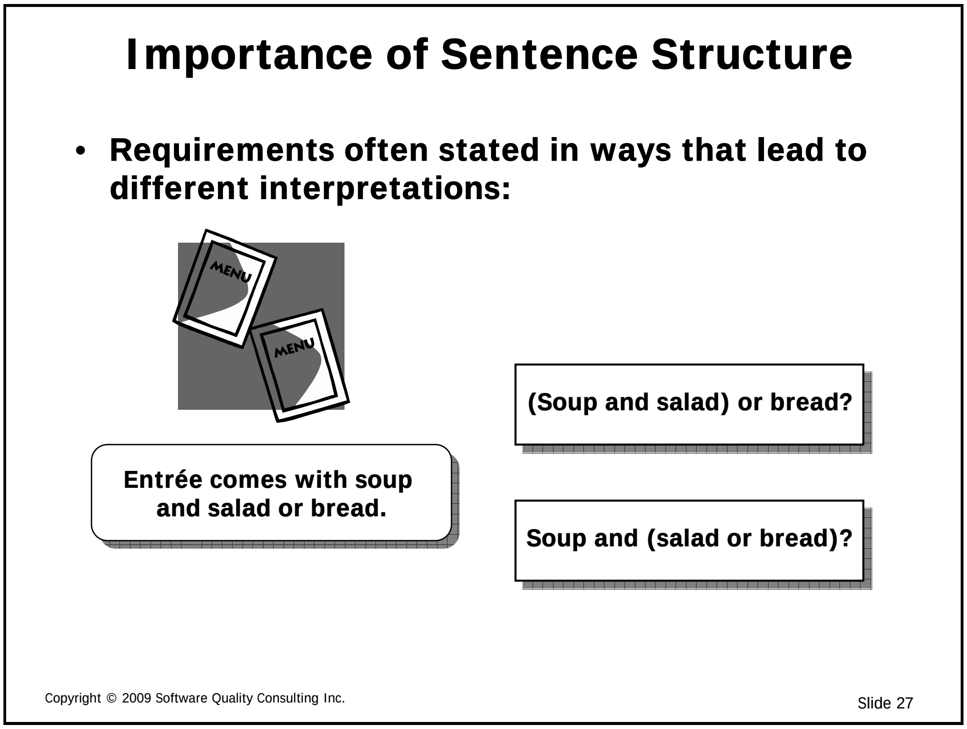


Importance of Sentence Structure

- **Use consistent sentence structure…**
- **Requirements from user perspective**
  - **User type:** “The [user class or actor name] …
  - **Expected Result:** … shall be able to [do something...]
  - **Object:** … [to something...]
  - **Qualifier:** [response time goal, quality objective]”
- **Example**
  - **The customer shall be able to add selected items to their shopping cart while browsing the web site.**

Wiegers, K. E., More About Software Requirements – Thorny Issues and Practical Advice, Microsoft Press, 2006

Slide 26



Importance of Sentence Structure

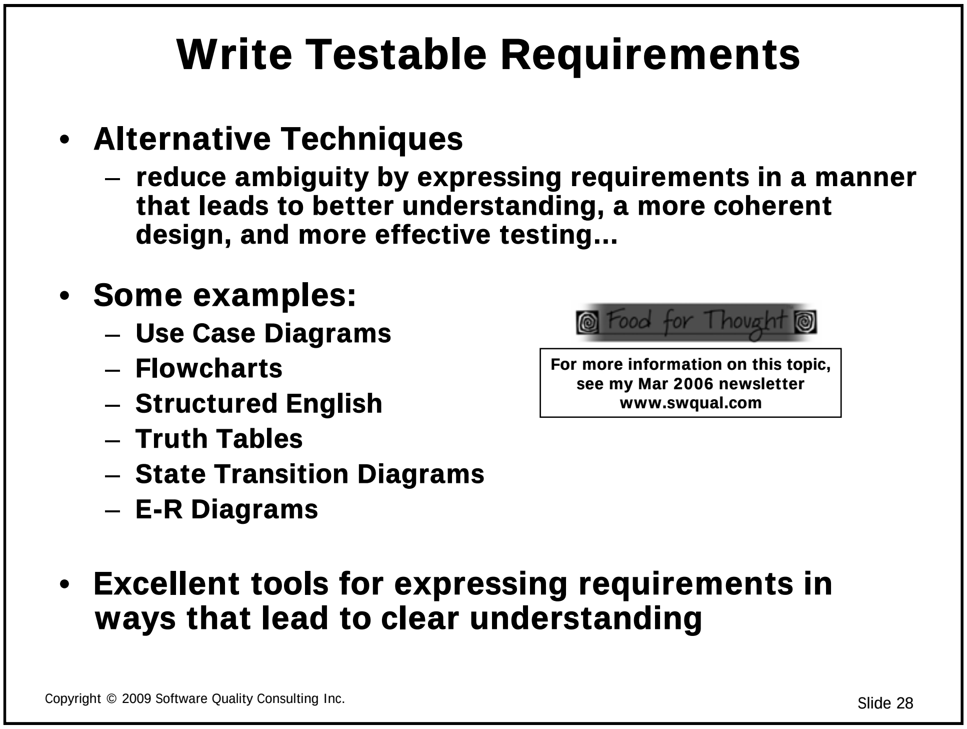
- Requirements often stated in ways that lead to different interpretations:

Entrée comes with soup and salad or bread.

(Soup and salad) or bread?

Soup and (salad or bread)?

Slide 27



Write Testable Requirements

- Alternative Techniques
  - reduce ambiguity by expressing requirements in a manner that leads to better understanding, a more coherent design, and more effective testing…
- Some examples:
  - Use Case Diagrams
  - Flowcharts
  - Structured English
  - Truth Tables
  - State Transition Diagrams
  - E-R Diagrams
- Excellent tools for expressing requirements in ways that lead to clear understanding

Food for Thought

For more information on this topic, see my Mar 2006 newsletter
www.swqual.com

Slide 28

Estimating and Scheduling Best Practices

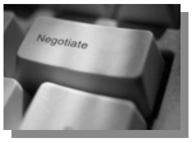
- **Estimates**
  - A **tentative evaluation** or **rough calculation**
  - Determined from experience doing similar tasks
  - Estimates are never negotiable…
  - Good estimates can only come from good requirements
- “A good estimate [...] provides a clear enough view of the project to allow the project leadership to make good decisions about how to control a project in order to hit its targets.”

McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

Slide 29

Estimating and Scheduling Best Practices

- **Targets**
  - Descriptions of desirable business objectives…
  - Determined by making business decisions…
  - Are internal to the business…
  - Targets are always negotiable…

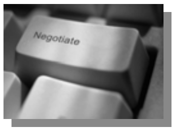


McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

Slide 30

Estimating and Scheduling Best Practices

- **Commitments**
  - Promises to deliver defined functionality at a specific level of quality by a specific date…
  - Promises made specifically to customers (internal or external)
  - Commitments are always negotiable…

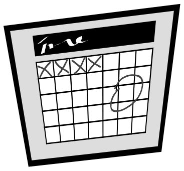


McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

Slide 31

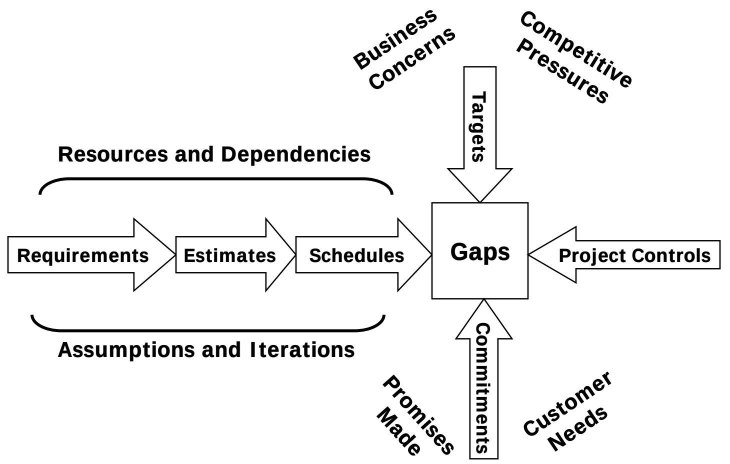
Estimating and Scheduling Best Practices

- **Schedules**
  - A schedule consists of a list of a project's terminal elements with intended start and finish dates
  - Terminal elements are items that are estimated in terms of resource requirements, budget and duration, linked by dependencies and scheduled
  - Schedules are always negotiable…
  - Good schedules can only come from good estimates…



Slide 32

Estimates, Targets, Commitments, and Schedules



Business Concerns

Competitive Pressures

Targets

Resources and Dependencies

Requirements

Estimates

Schedules

Gaps

Project Controls

Assumptions and Iterations

Commitments

Promises Made

Customer Needs

McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

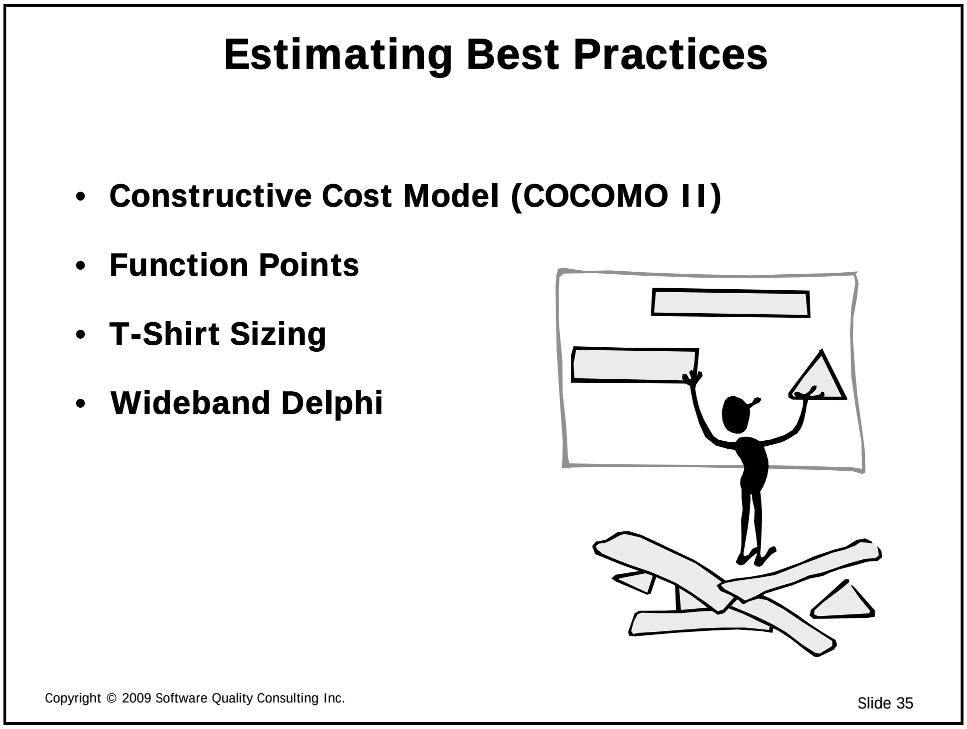
Slide 33

Estimates, Targets, Commitments, and Schedules

- **Some Problems**
  - **When Managers asks for estimates often interested in something else…**
  - **Characteristics of many estimates:**
    - **Derived using “gut feel”, “finger in the wind” methods**
    - **Not supported by data or factual information**
    - **Answer boss wants to hear**
  - **“It is difficult to make a vigorous, plausible, and job-risking defense of an estimate that is derived by no quantitative method, supported by little data, and certified chiefly by the hunches of managers”.**

Brooks, F., The Mythical Man-Month, 25th Anniversary Edition, Addison-Wesley, 1995

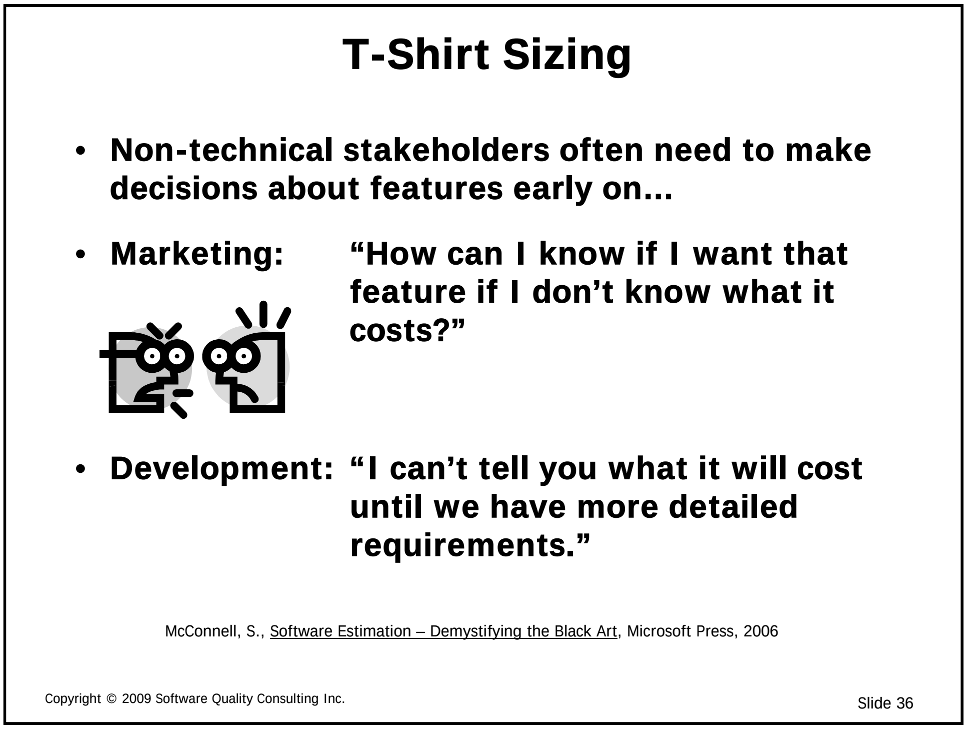
Slide 34



Estimating Best Practices

- Constructive Cost Model (COCOMO II)
- Function Points
- T-Shirt Sizing
- Wideband Delphi

Slide 35

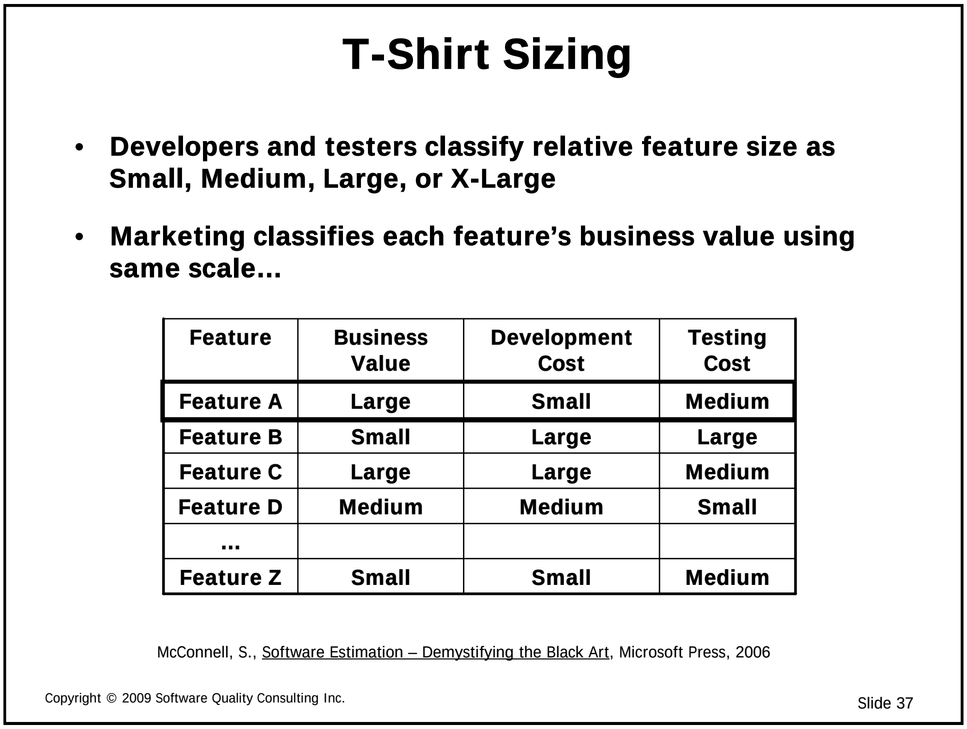


T-Shirt Sizing

- Non-technical stakeholders often need to make decisions about features early on…
- Marketing: “How can I know if I want that feature if I don’t know what it costs?”
- Development: “I can’t tell you what it will cost until we have more detailed requirements.”

McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

Slide 36



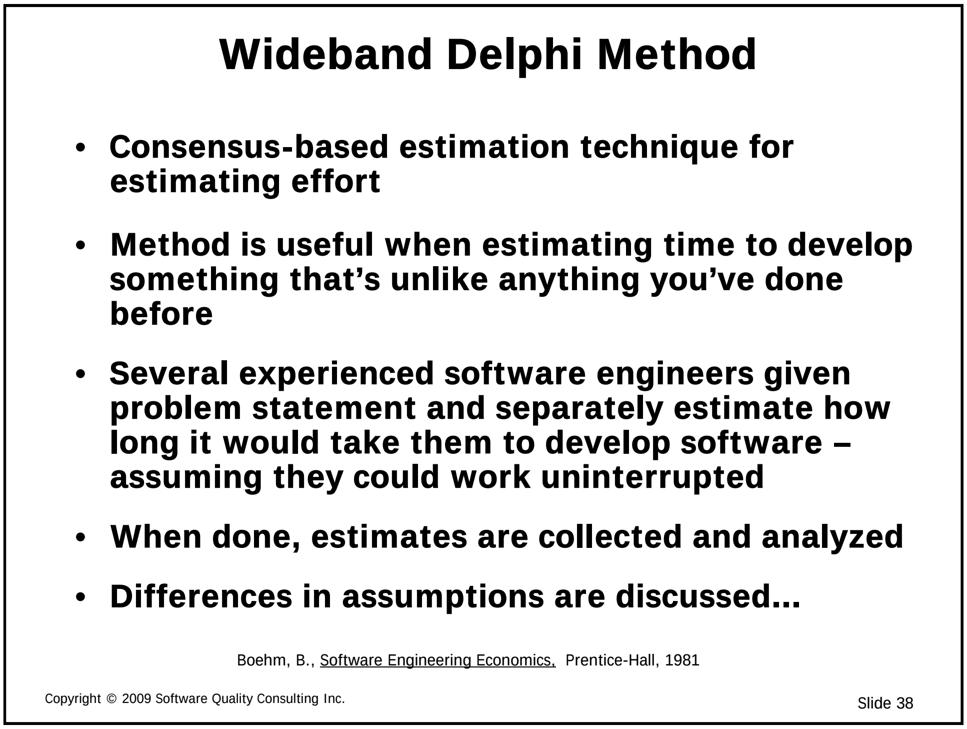
T-Shirt Sizing

- Developers and testers classify relative feature size as Small, Medium, Large, or X-Large
- Marketing classifies each feature's business value using same scale...

Feature	Business Value	Development Cost	Testing Cost
Feature A	Large	Small	Medium
Feature B	Small	Large	Large
Feature C	Large	Large	Medium
Feature D	Medium	Medium	Small
...			
Feature Z	Small	Small	Medium

McConnell, S., Software Estimation – Demystifying the Black Art, Microsoft Press, 2006

Slide 37

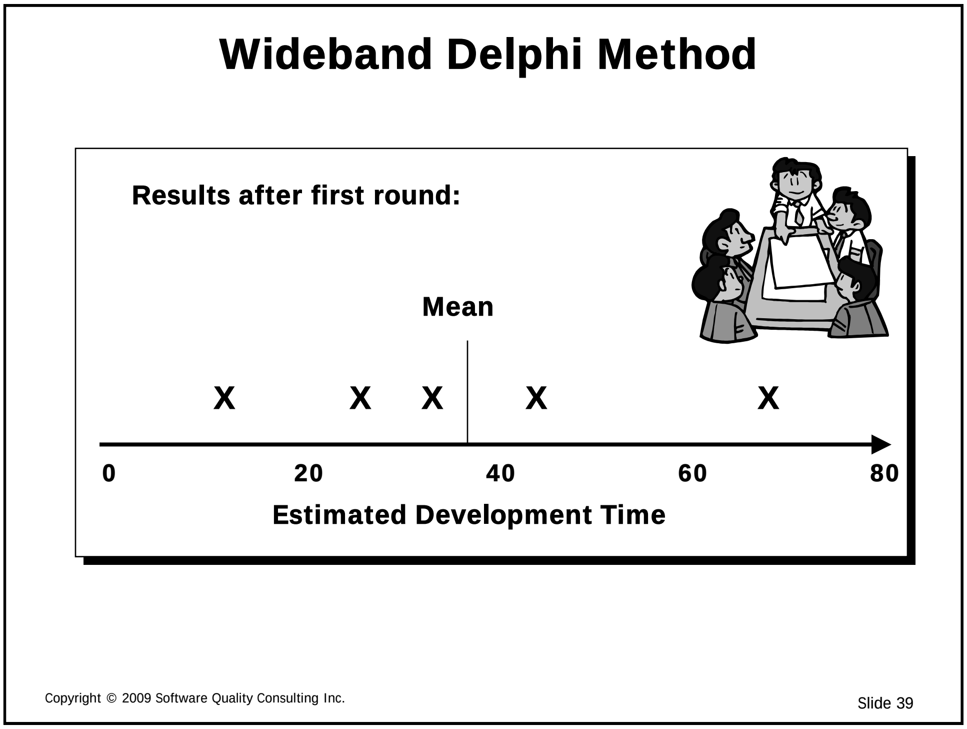


Wideband Delphi Method

- Consensus-based estimation technique for estimating effort
- Method is useful when estimating time to develop something that's unlike anything you've done before
- Several experienced software engineers given problem statement and separately estimate how long it would take them to develop software – assuming they could work uninterrupted
- When done, estimates are collected and analyzed
- Differences in assumptions are discussed...

Boehm, B., Software Engineering Economics, Prentice-Hall, 1981

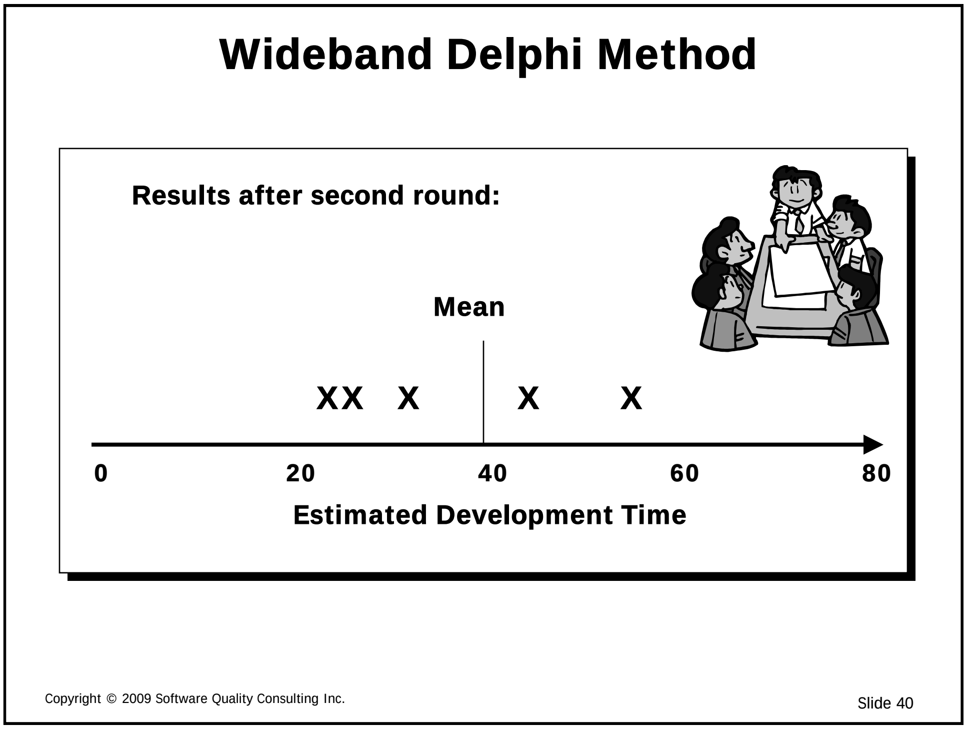
Slide 38



Wideband Delphi Method

Results after first round:

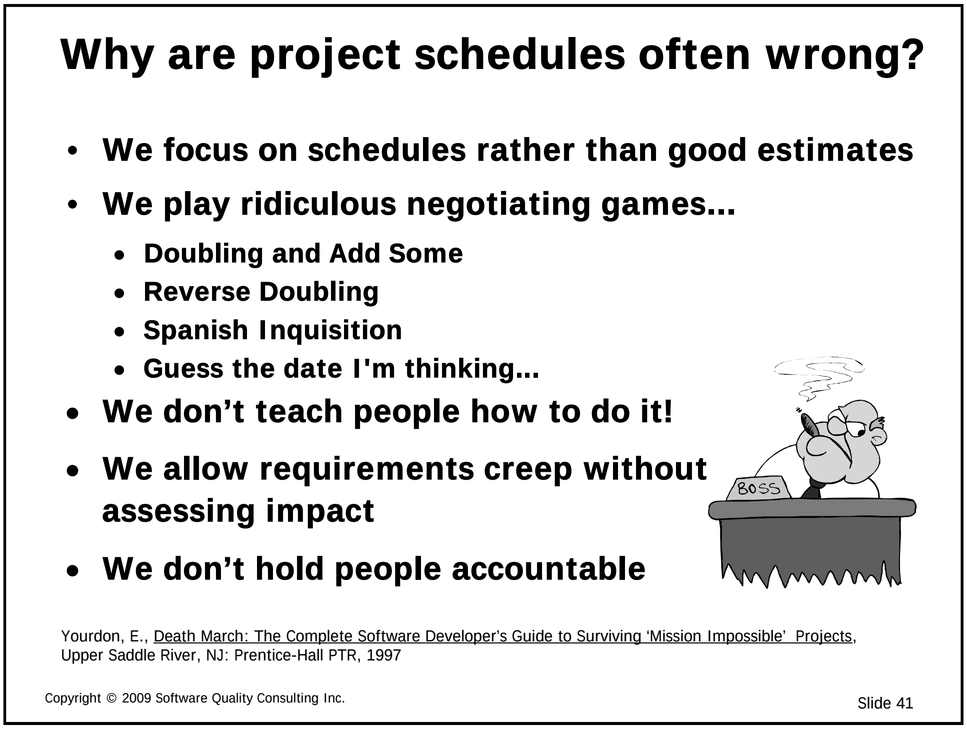
Slide 39



Wideband Delphi Method

Results after second round:

Slide 40

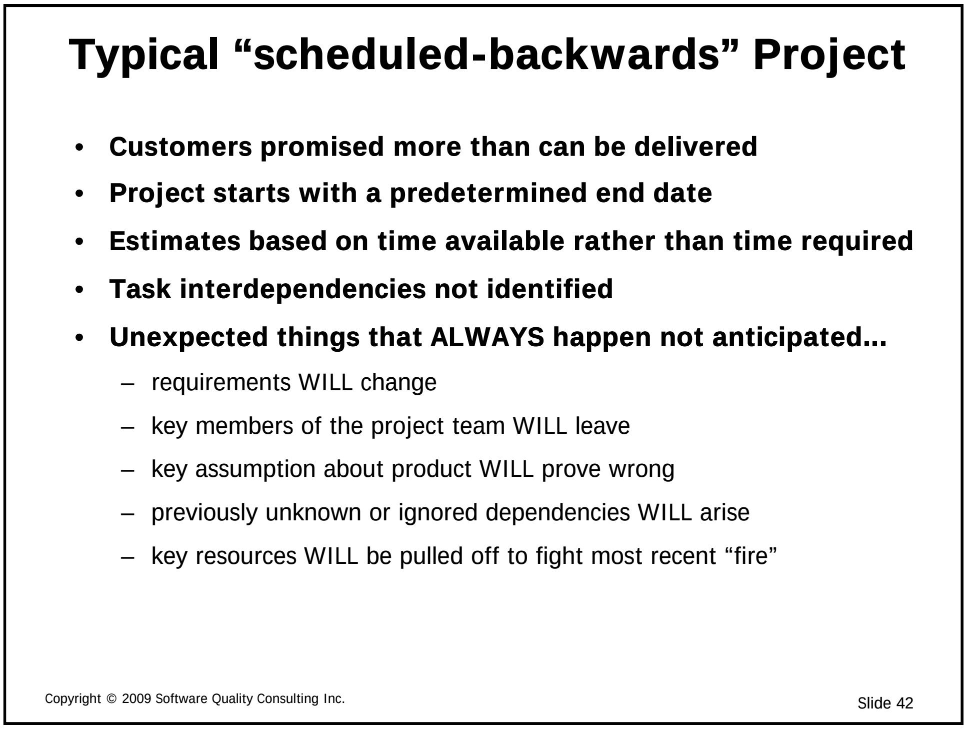


Why are project schedules often wrong?

- We focus on schedules rather than good estimates
- We play ridiculous negotiating games...
  - Doubling and Add Some
  - Reverse Doubling
  - Spanish Inquisition
  - Guess the date I'm thinking...
- We don’t teach people how to do it!
- We allow requirements creep without assessing impact
- We don’t hold people accountable

Yourdon, E., Death March: The Complete Software Developer’s Guide to Surviving ‘Mission Impossible’ Projects, Upper Saddle River, NJ: Prentice-Hall PTR, 1997

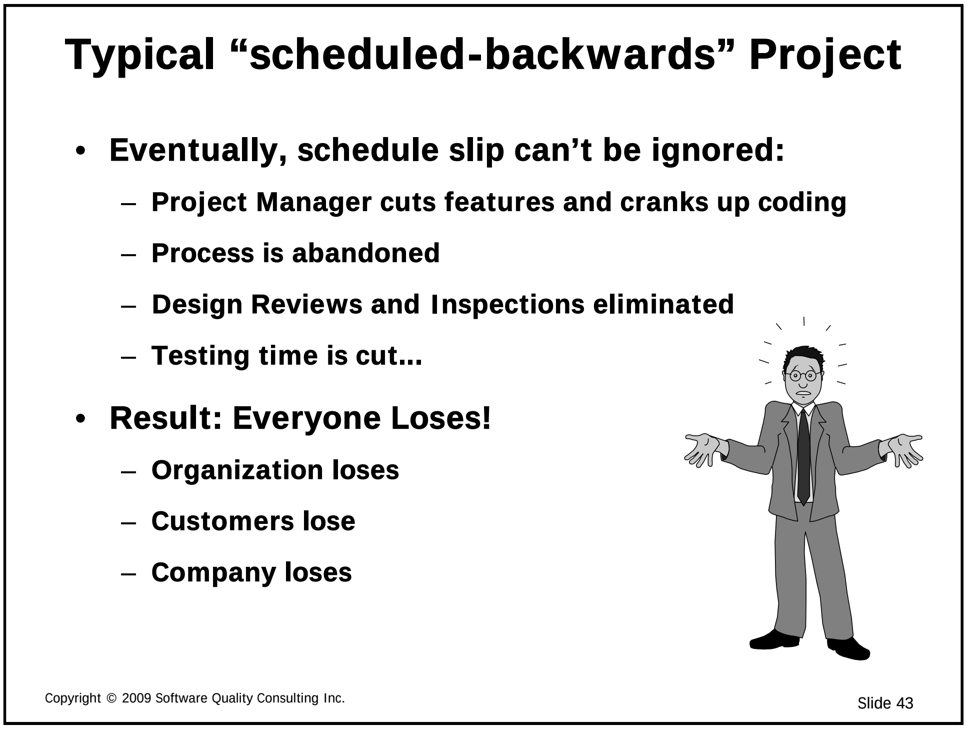
Slide 41



Typical “scheduled-backwards” Project

- Customers promised more than can be delivered
- Project starts with a predetermined end date
- Estimates based on time available rather than time required
- Task interdependencies not identified
- Unexpected things that ALWAYS happen not anticipated...
  - requirements WILL change
  - key members of the project team WILL leave
  - key assumption about product WILL prove wrong
  - previously unknown or ignored dependencies WILL arise
  - key resources WILL be pulled off to fight most recent “fire”

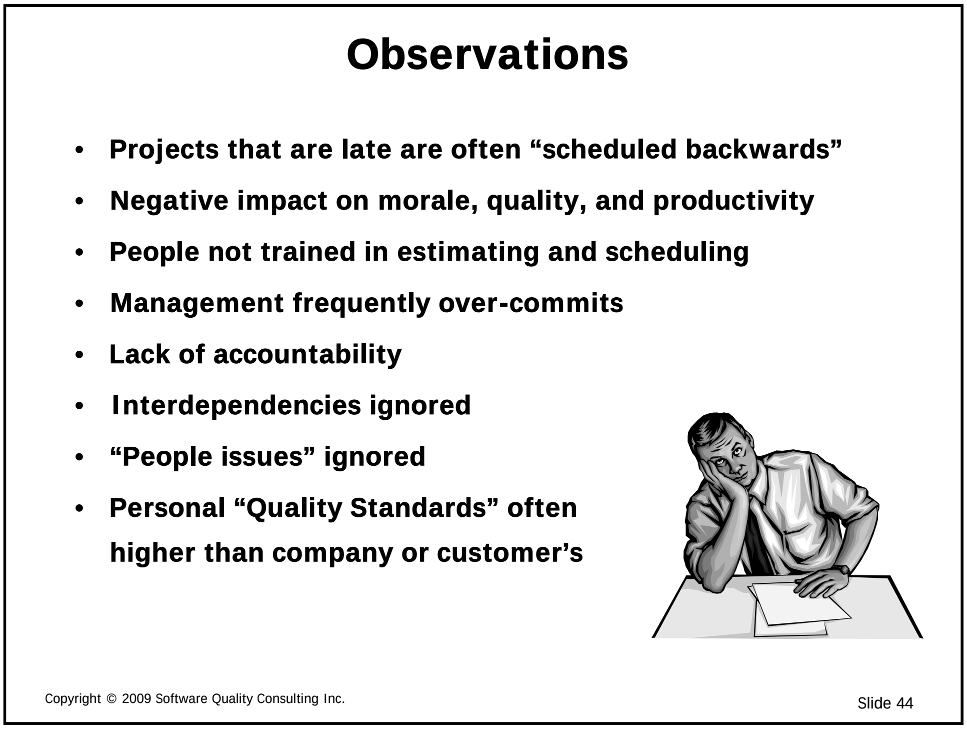
Slide 42



Typical “scheduled-backwards” Project

- Eventually, schedule slip can’t be ignored:
  - Project Manager cuts features and cranks up coding
  - Process is abandoned
  - Design Reviews and Inspections eliminated
  - Testing time is cut…
- Result: Everyone Loses!
  - Organization loses
  - Customers lose
  - Company loses

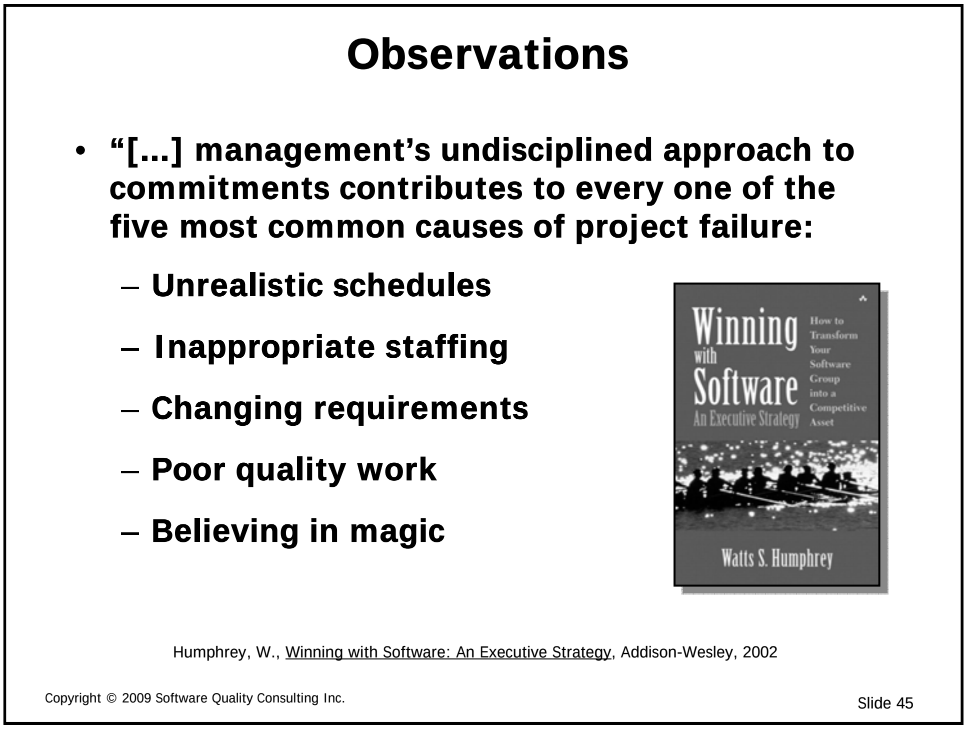
Slide 43



Observations

- Projects that are late are often “scheduled backwards”
- Negative impact on morale, quality, and productivity
- People not trained in estimating and scheduling
- Management frequently over-commits
- Lack of accountability
- Interdependencies ignored
- “People issues” ignored
- Personal “Quality Standards” often higher than company or customer’s

Slide 44



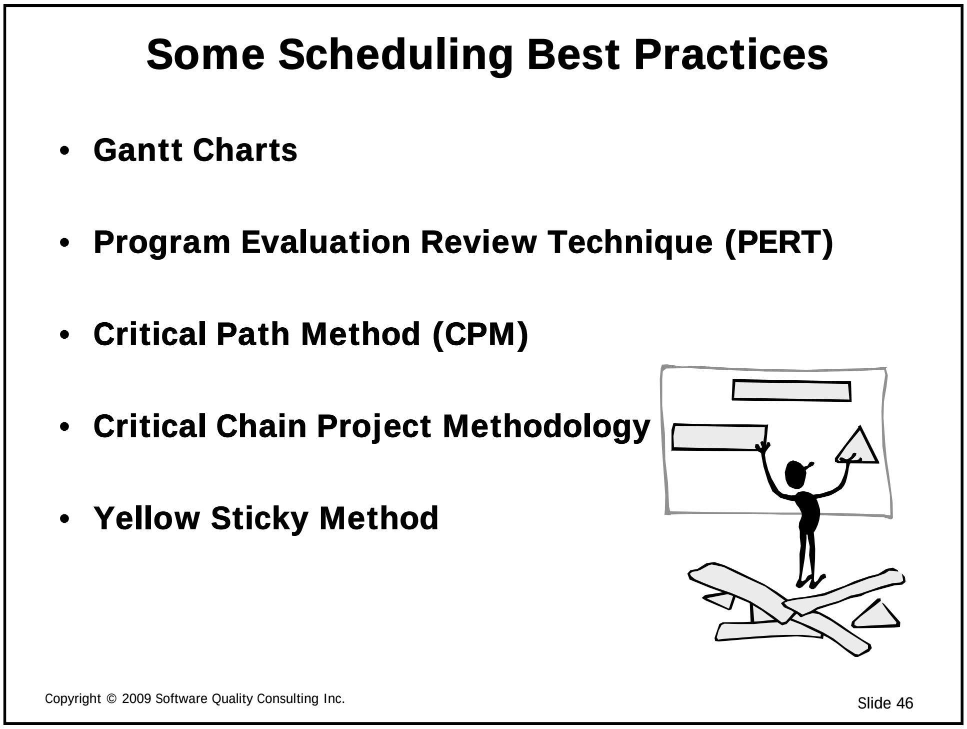
Observations

- **“[...] management’s undisciplined approach to commitments contributes to every one of the five most common causes of project failure:**
  - **Unrealistic schedules**
  - **Inappropriate staffing**
  - **Changing requirements**
  - **Poor quality work**
  - **Believing in magic**

Winning with Software
An Executive Strategy
How to Transform Your Software Group into a Competitive Asset
Watts S. Humphrey

Humphrey, W., Winning with Software: An Executive Strategy, Addison-Wesley, 2002

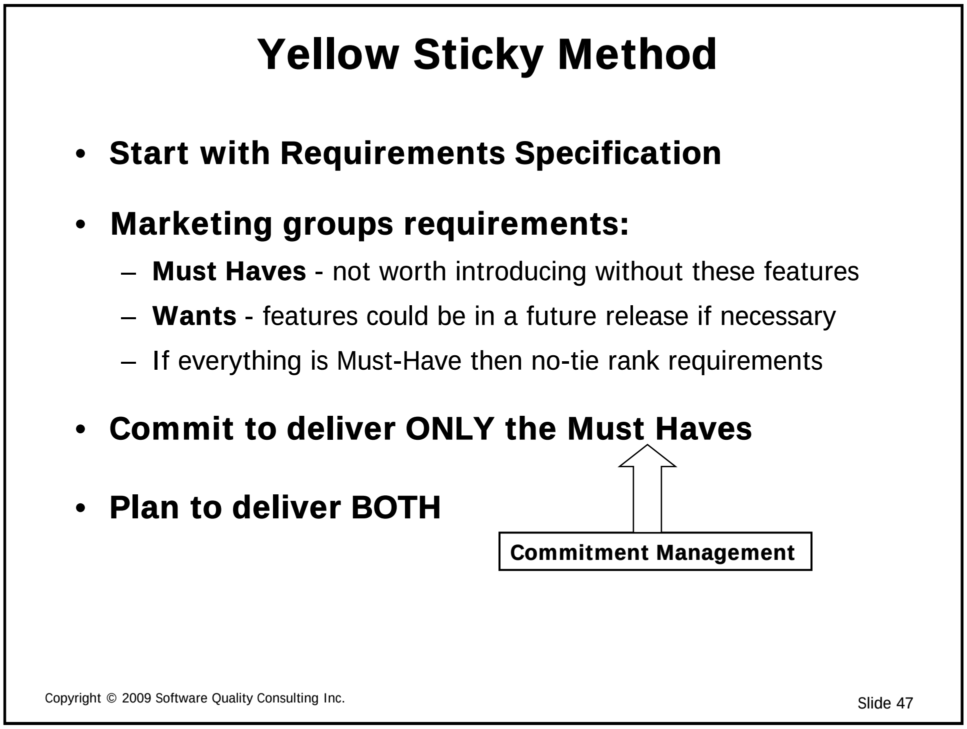
Slide 45



Some Scheduling Best Practices

- **Gantt Charts**
- **Program Evaluation Review Technique (PERT)**
- **Critical Path Method (CPM)**
- **Critical Chain Project Methodology**
- **Yellow Sticky Method**

Slide 46

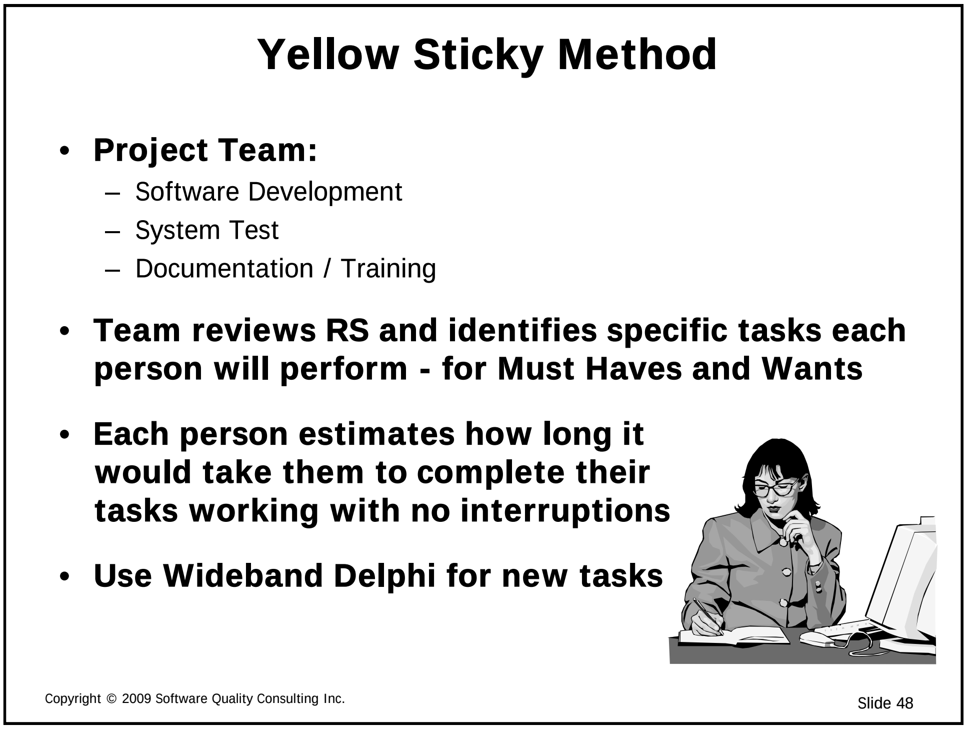


Yellow Sticky Method

- **Start with Requirements Specification**
- **Marketing groups requirements:**
  - **Must Haves** - not worth introducing without these features
  - **Wants** - features could be in a future release if necessary
  - If everything is Must-Have then no-tie rank requirements
- **Commit to deliver ONLY the Must Haves**
- **Plan to deliver BOTH**

Commitment Management

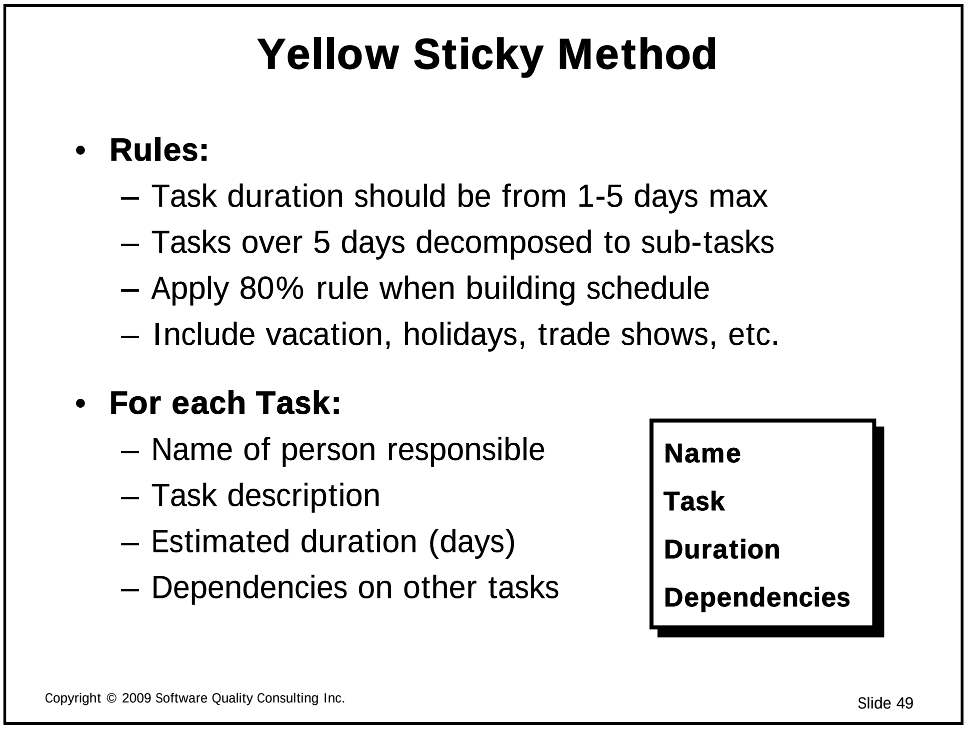
Slide 47



Yellow Sticky Method

- **Project Team:**
  - Software Development
  - System Test
  - Documentation / Training
- **Team reviews RS and identifies specific tasks each person will perform - for Must Haves and Wants**
- **Each person estimates how long it would take them to complete their tasks working with no interruptions**
- **Use Wideband Delphi for new tasks**

Slide 48

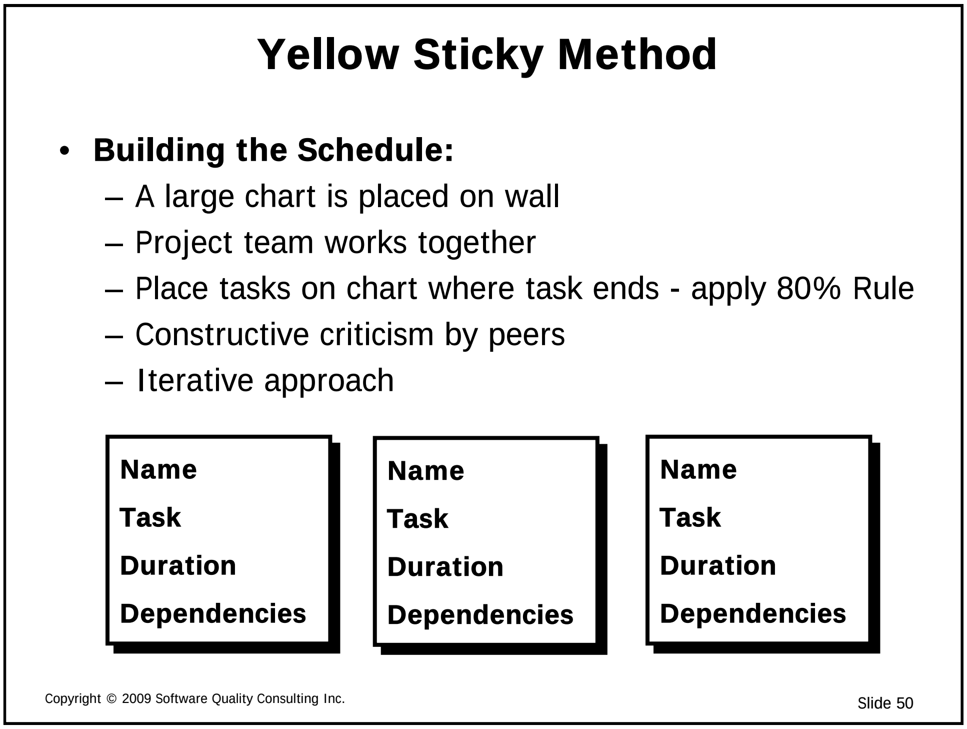


Yellow Sticky Method

- **Rules:**
  - – Task duration should be from 1-5 days max
  - – Tasks over 5 days decomposed to sub-tasks
  - – Apply 80% rule when building schedule
  - – Include vacation, holidays, trade shows, etc.
- **For each Task:**
  - – Name of person responsible
  - – Task description
  - – Estimated duration (days)
  - – Dependencies on other tasks

Name
Task
Duration
Dependencies

Slide 49



Yellow Sticky Method

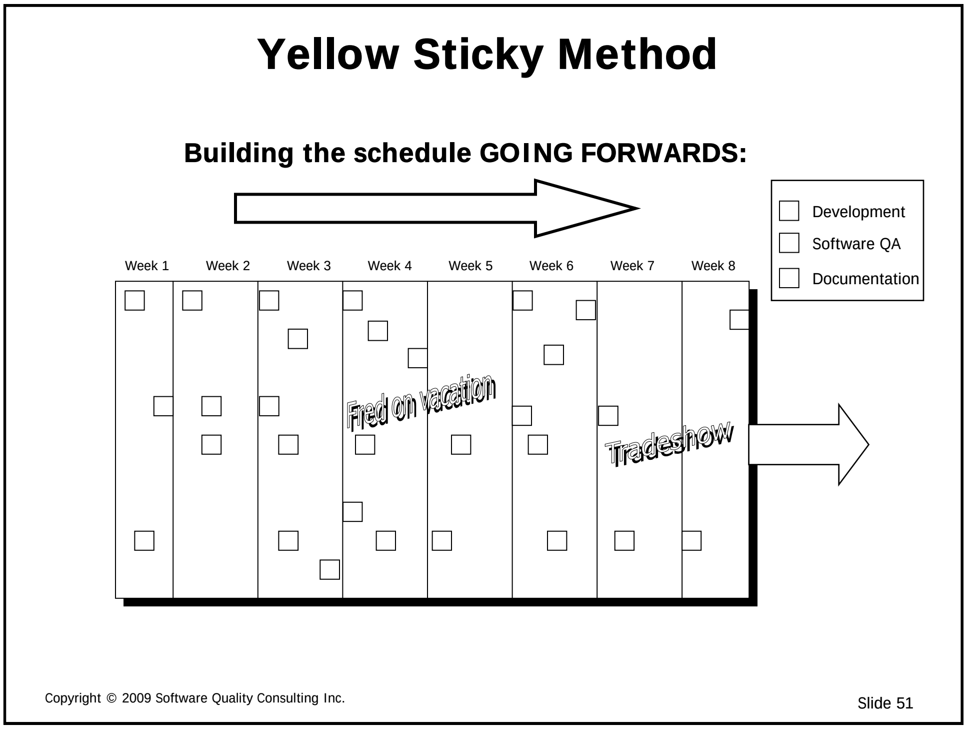
- **Building the Schedule:**
  - – A large chart is placed on wall
  - – Project team works together
  - – Place tasks on chart where task ends - apply 80% Rule
  - – Constructive criticism by peers
  - – Iterative approach

Name
Task
Duration
Dependencies

Name
Task
Duration
Dependencies

Name
Task
Duration
Dependencies

Slide 50



Yellow Sticky Method

Building the schedule GOING FORWARDS:

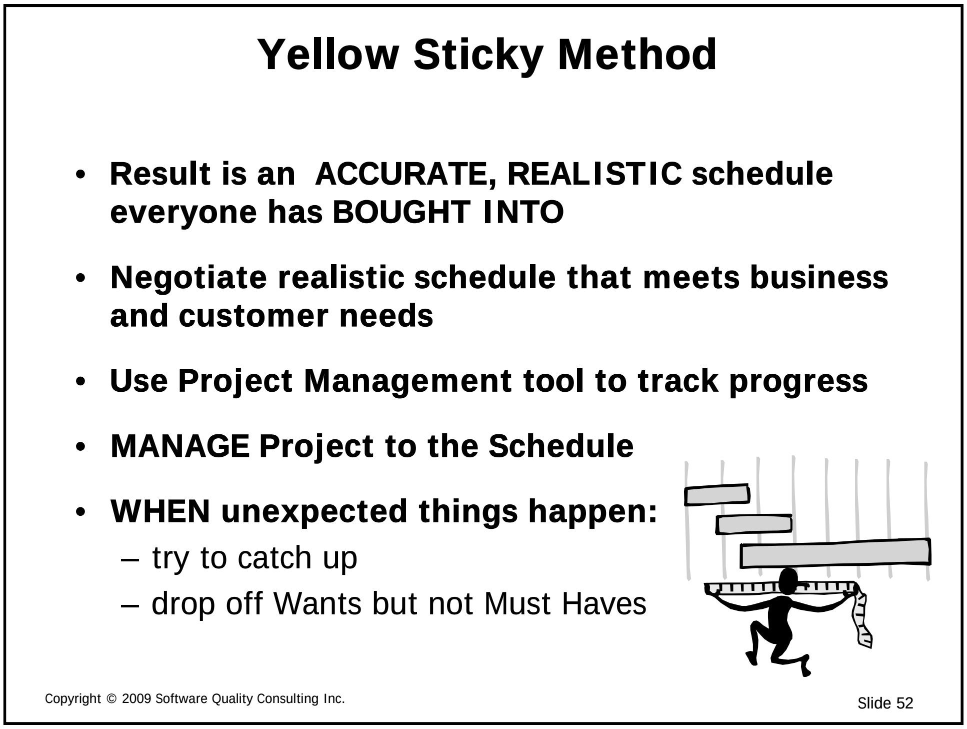
□ Development
□ Software QA
□ Documentation

Week 1 Week 2 Week 3 Week 4 Week 5 Week 6 Week 7 Week 8

Fred on vacation

Tradeshow

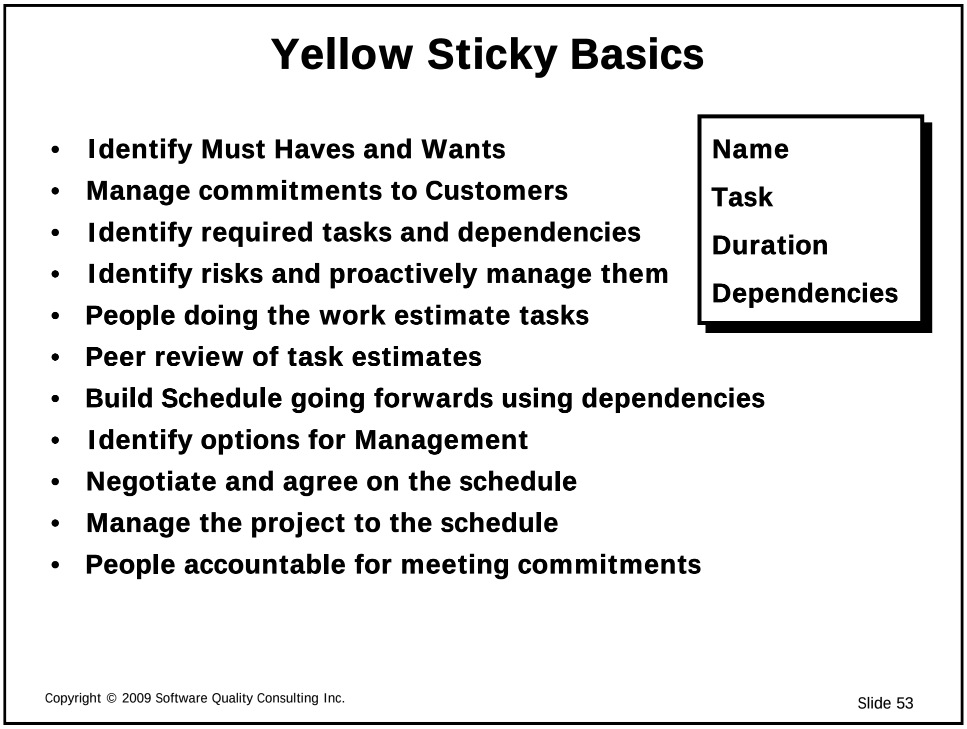
Slide 51



Yellow Sticky Method

- **Result is an ACCURATE, REALISTIC schedule everyone has BOUGHT INTO**
- **Negotiate realistic schedule that meets business and customer needs**
- **Use Project Management tool to track progress**
- **MANAGE Project to the Schedule**
- **WHEN unexpected things happen:**
  - try to catch up
  - drop off Wants but not Must Haves

Slide 52

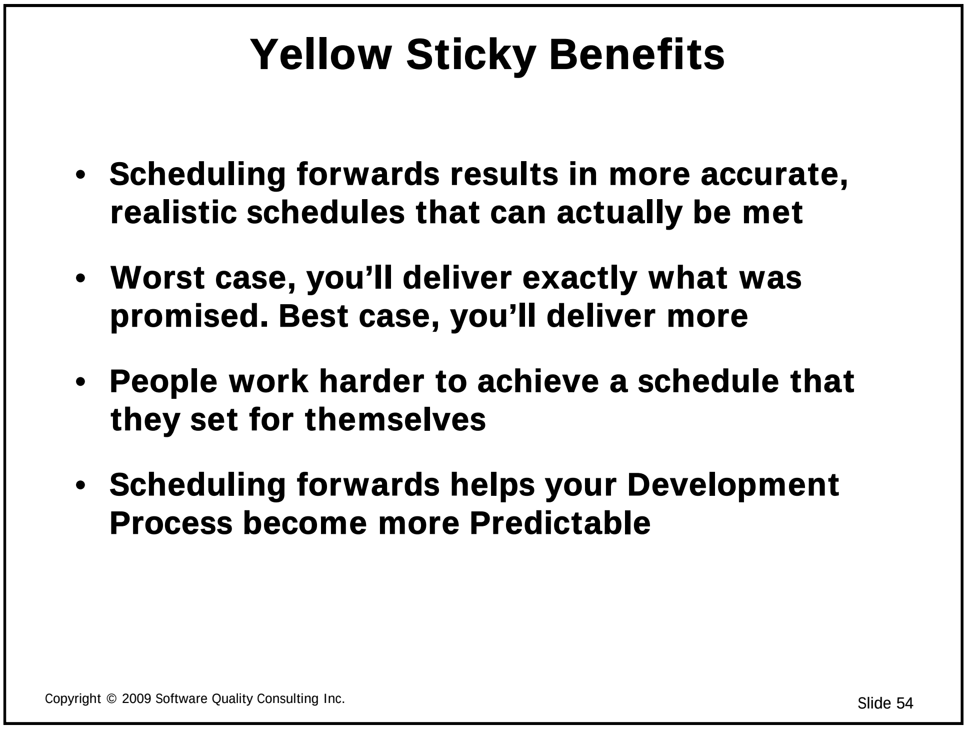


Yellow Sticky Basics

- Identify Must Haves and Wants
- Manage commitments to Customers
- Identify required tasks and dependencies
- Identify risks and proactively manage them
- People doing the work estimate tasks
- Peer review of task estimates
- Build Schedule going forwards using dependencies
- Identify options for Management
- Negotiate and agree on the schedule
- Manage the project to the schedule
- People accountable for meeting commitments

Name
Task
Duration
Dependencies

Slide 53



Yellow Sticky Benefits

- Scheduling forwards results in more accurate, realistic schedules that can actually be met
- Worst case, you’ll deliver exactly what was promised. Best case, you’ll deliver more
- People work harder to achieve a schedule that they set for themselves
- Scheduling forwards helps your Development Process become more Predictable

Slide 54

How to know if you’re on right track?

- **Software Development Process**
  - **Is it well-defined and is it followed?**
  - **Does it have objective criteria for measuring progress?**
- **Estimating**
  - **Provide training in basic estimating techniques…**
  - **Track estimates and compare to actuals, reconcile differences**
- **Scheduling**
  - **Track schedules to original commitments, reconcile differences**
- **Product Quality**
  - **Defect Removal Efficiency…**
  - **Is number of unplanned bug fix releases declining?**
- **Commitments and Risk**
  - **Strive to under-commit and over-deliver**
  - **Ensure project managers are focused on risk**

Slide 55

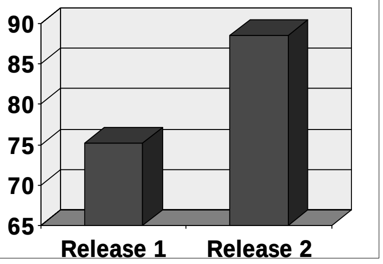
Measuring Software Quality: Post Release

- **Defect Removal Efficiency**

$$\frac{\textbf{Number defects found during development}}{\textbf{Number found during development + Number reported by Customers}}$$

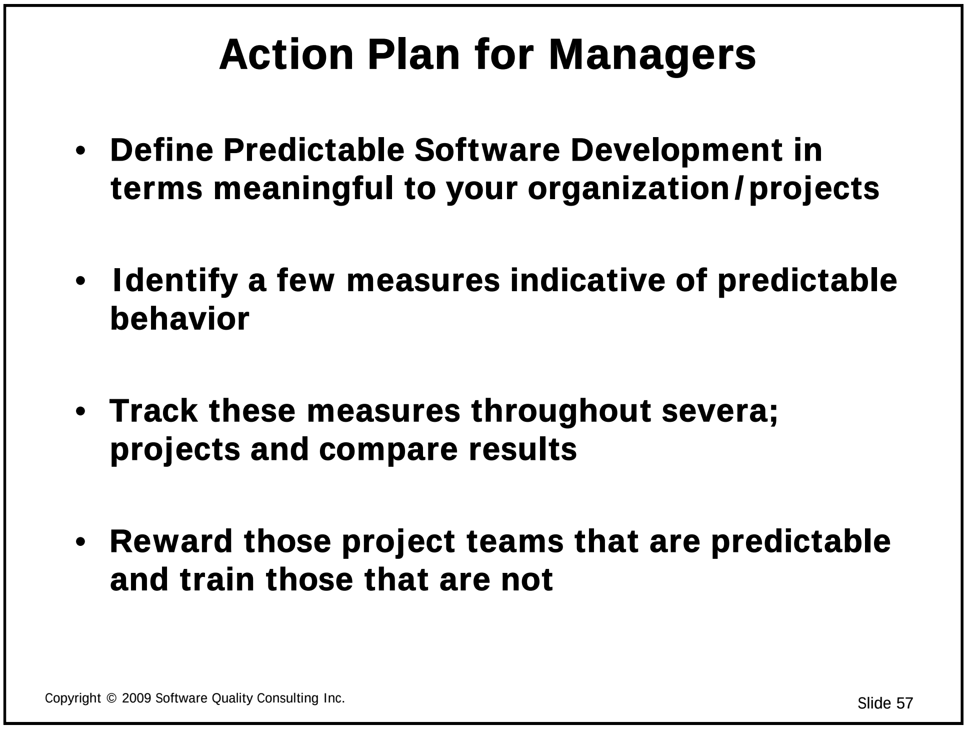
based on at least n months of Customer use

- Average for most software: 75-85%
- Best in Class: 99.5% or better



Jones, C., “Software Defect-removal Efficiency”, *IEEE Computer*, Vol. 29, No. 4, April 1996, pp. 94-95.

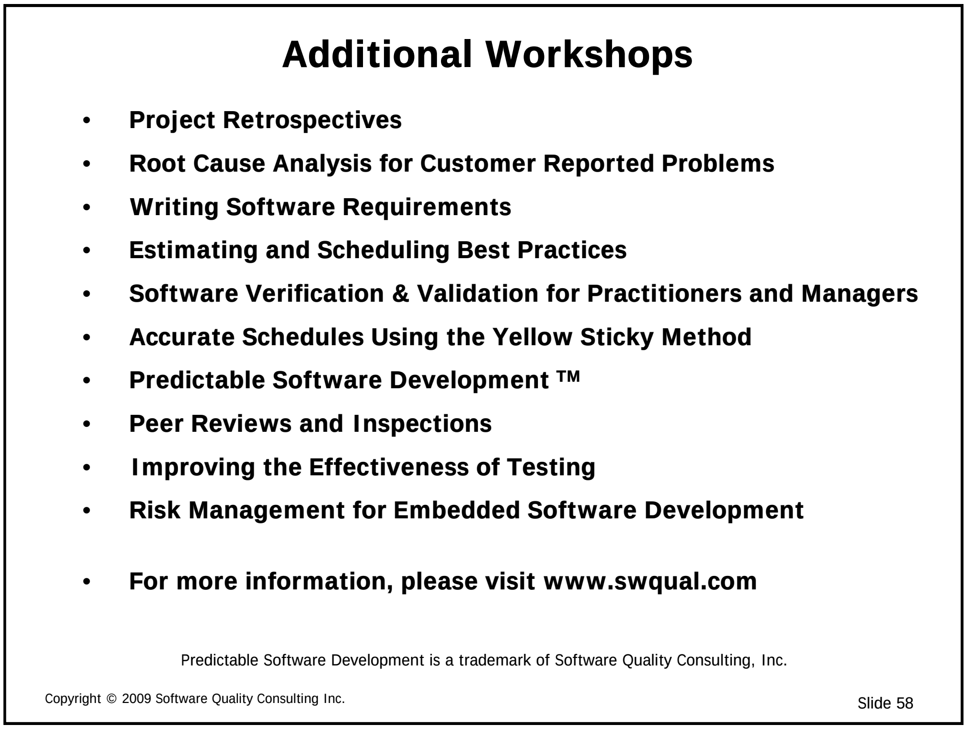
Slide 56



Action Plan for Managers

- Define Predictable Software Development in terms meaningful to your organization/projects
- Identify a few measures indicative of predictable behavior
- Track these measures throughout severa; projects and compare results
- Reward those project teams that are predictable and train those that are not

Slide 57



Additional Workshops

- Project Retrospectives
- Root Cause Analysis for Customer Reported Problems
- Writing Software Requirements
- Estimating and Scheduling Best Practices
- Software Verification & Validation for Practitioners and Managers
- Accurate Schedules Using the Yellow Sticky Method
- Predictable Software Development ™
- Peer Reviews and Inspections
- Improving the Effectiveness of Testing
- Risk Management for Embedded Software Development

- For more information, please visit www.swqual.com

Predictable Software Development is a trademark of Software Quality Consulting, Inc.

Slide 58

Thank you...

- If you have questions, please call or e-mail...
- To opt-in to my e-newsletter and view past newsletters, visit www.swqual.com

Food for Thought™

The e-newsletter for Software Development and QA Professionals

Software Quality Consulting Inc.

Steven R. Rakitin
President

- Consulting
- Training
- Auditing

Phone: 508.529.4282
Fax: 508.529.7799

www.swqual.com
steve@swqual.com