The Art & Science of Software Process

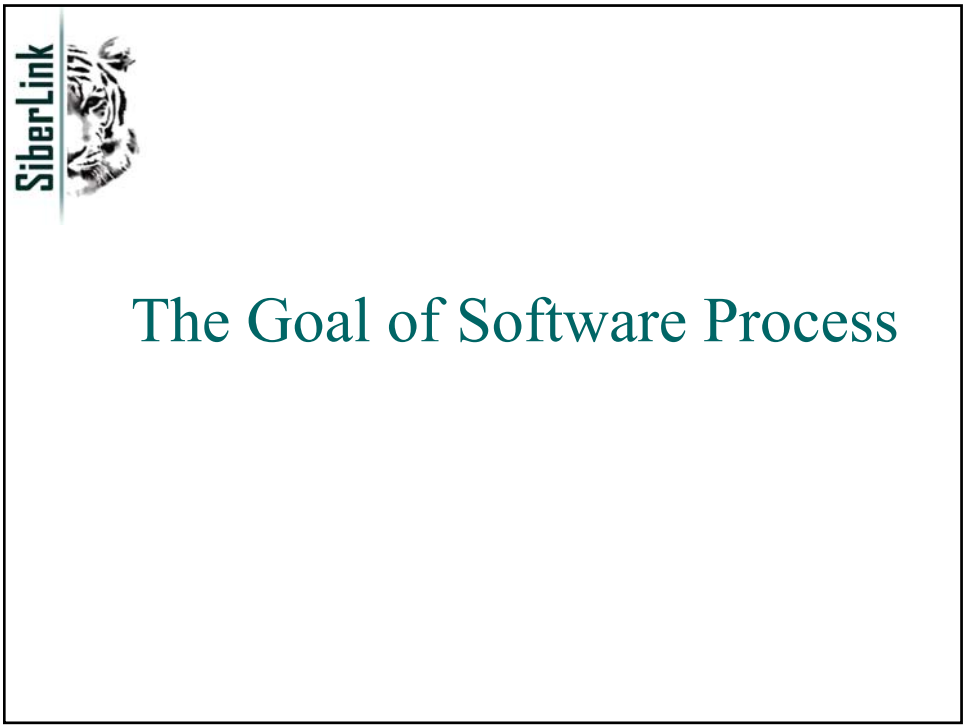
Steven Teleki
Vice President, Software Engineering, SiberLink, Inc.
Chairman, IEEE Computer Society, Austin Chapter

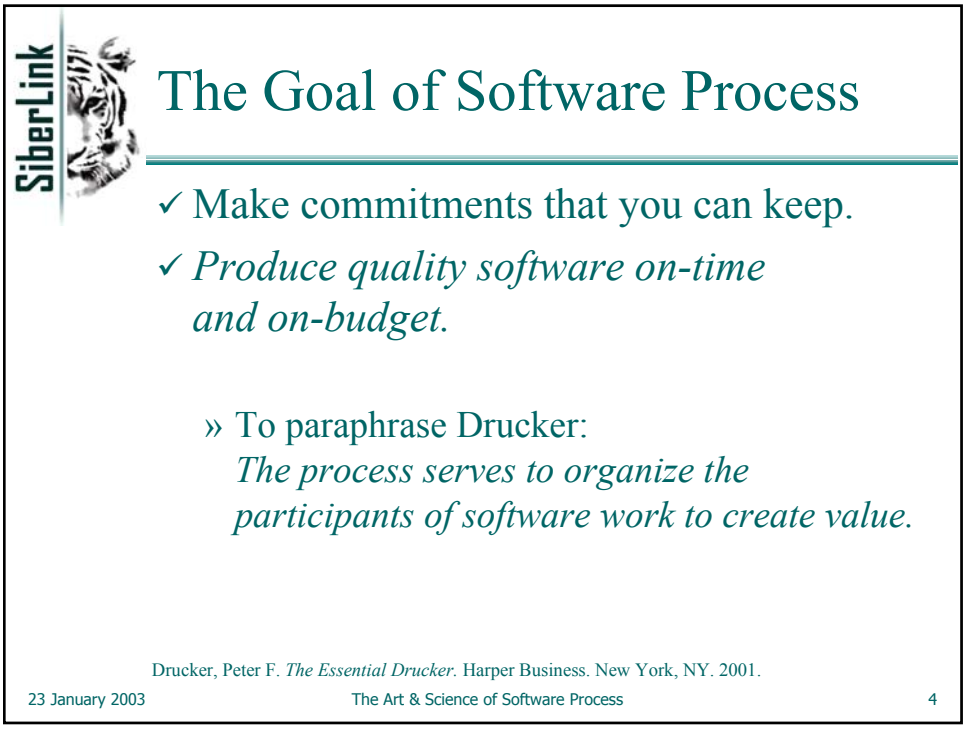
Why *Art & Science*?

When asked why he gave the title, *The Art of Computer Programming,* to his famous series of books, Donald Knuth said:

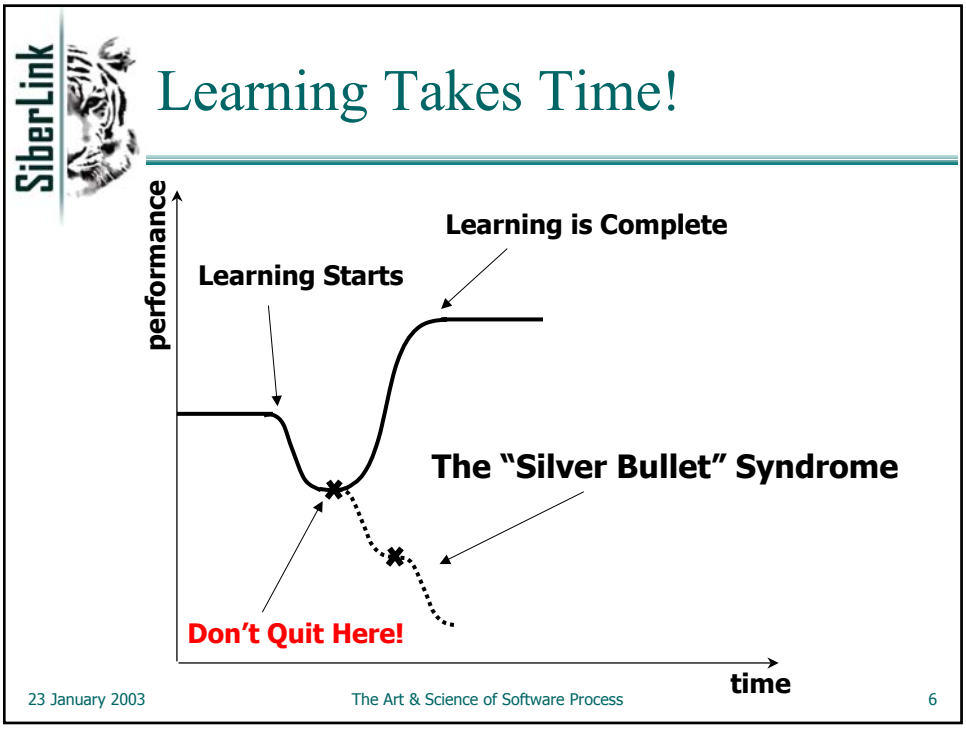
"Science is what we understand well enough to explain to a computer and art is everything else."

Knuth, Donald. *Computer Programming is an Art.* Communications of the ACM. December 1974.


SiberLink
The Goal of Software Process


SiberLink
The Goal of Software Process
✓ Make commitments that you can keep.
✓ Produce quality software on-time and on-budget.
» To paraphrase Drucker:
The process serves to organize the participants of software work to create value.
Drucker, Peter F. The Essential Drucker. Harper Business. New York, NY. 2001.
23 January 2003
The Art & Science of Software Process
4


SiberLink
Challenges


SiberLink
Learning Takes Time!
performance
Learning is Complete
Learning Starts
The "Silver Bullet" Syndrome
Don't Quit Here!
time
23 January 2003
The Art & Science of Software Process
6

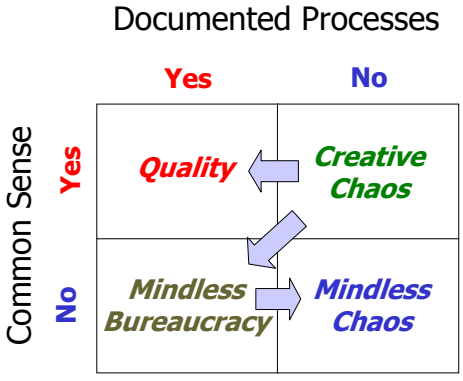
Is Learning Difficult?

- ✓ Crawl, walk, run!
  - » An accomplished walker doesn't think about the mechanics of the steps anymore.
- ✓ *Learning dilemma:*

  We learn best from experience but we never directly experience the consequences of many of our most important decisions.

Senge, Peter. *The Fifth Discipline.* Pg. 23. Currency Doubleday. New York, NY. 1990.

You'll Need Common Sense!



From Mark Paulk, with thanks to Sanjiv Ahuja, President and COO of Bellcore.

Execution vs. Enactment

- ✓ Execution: carrying out a process without much thinking or judgment.
  "Unencumbered by the thought process."
  - » A computer executes a program.
- ✓ Enactment: carrying out a process with understanding of each step and using the process as a guide.
  "If the map and the terrain don't match, trust the terrain."

Thanks to Click & Clack, the CarTalk guys on NPR.

Everything Seems Crazy at First!

"We should do something when people say it is crazy. If people say something is 'good,' it means someone else is already doing it."

» Hajime Mitarai, president, Canon

Peters, Thomas J. *The Circle of Innovation, You Can't Shrink Your Way To Greatness*. Vintage Books. New York, NY, 1997.

Organizational Expectations: What Changed In 140+ Years?

"Wanted: Young, skinny, wiry fellows not over 18. Must be expert riders willing to risk death daily. Orphans preferred. Wages $25 per week."

- Pony Express advertisement, 1860.

"We realize the skills, intellect and personality we seek are rare, and our compensation plan reflects that. In return we expect TOTAL AND ABSOLUTE COMMITMENT to project success—overcoming all obstacles to create applications on time and within budget."

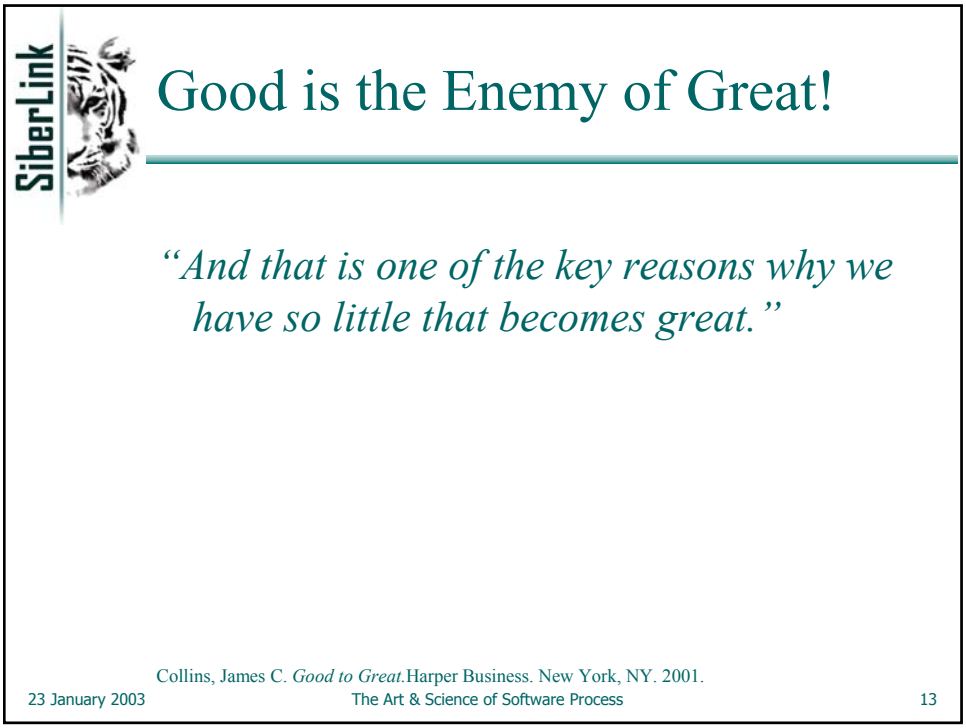
- Software Developer Advertisement, Seattle Times, 1995.

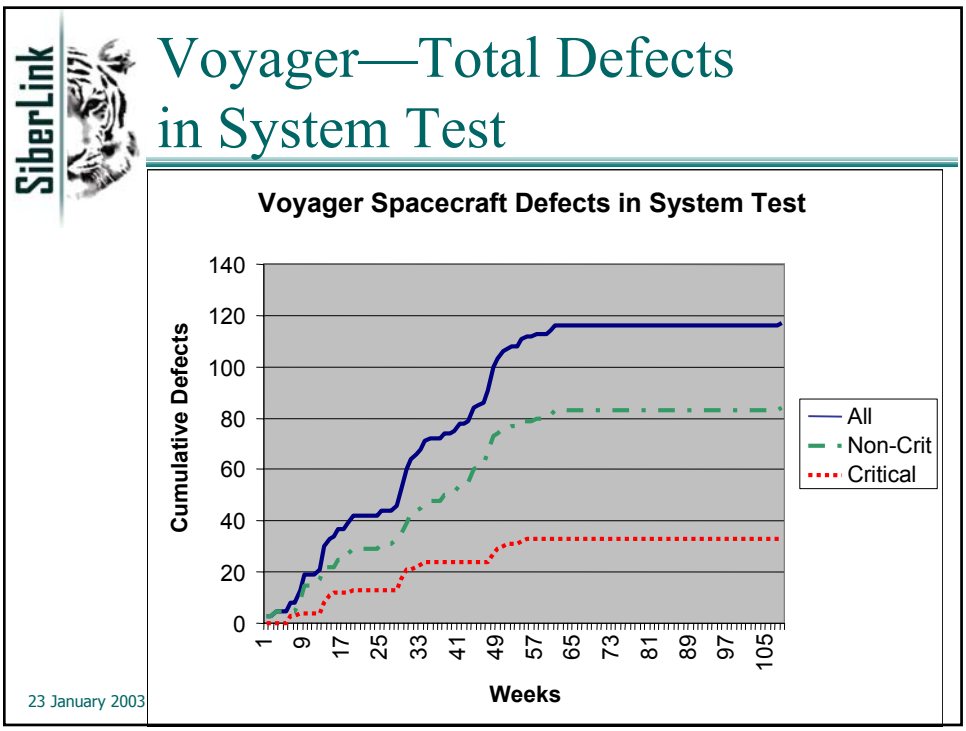
McConnell, Steve. *After the Gold Rush*. Microsoft Press. 1999.

We Must Cope With Ignorance

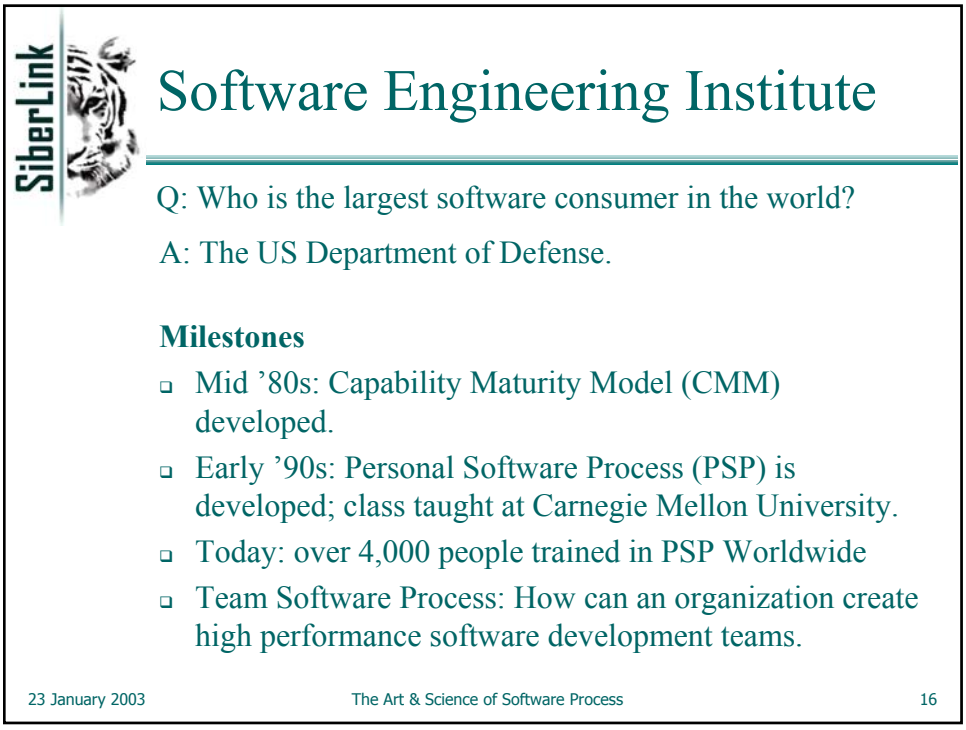
- ✓ 0th Order of Ignorance: Lack of Ignorance. You know.
- ✓ 1st Order of Ignorance: Lack of knowledge. You know the question.
- ✓ 2nd Order of Ignorance: Lack of awareness. This is a real problem: not only you don't know the answer, you don't even know what the question is.
- ✓ 3rd Order of Ignorance: Lack of process. You don't have a process to find out what it is that you don't know.
- ✓ 4th Order of Ignorance: Meta Ignorance. You don't know about the orders of ignorance. You are past this. ☺

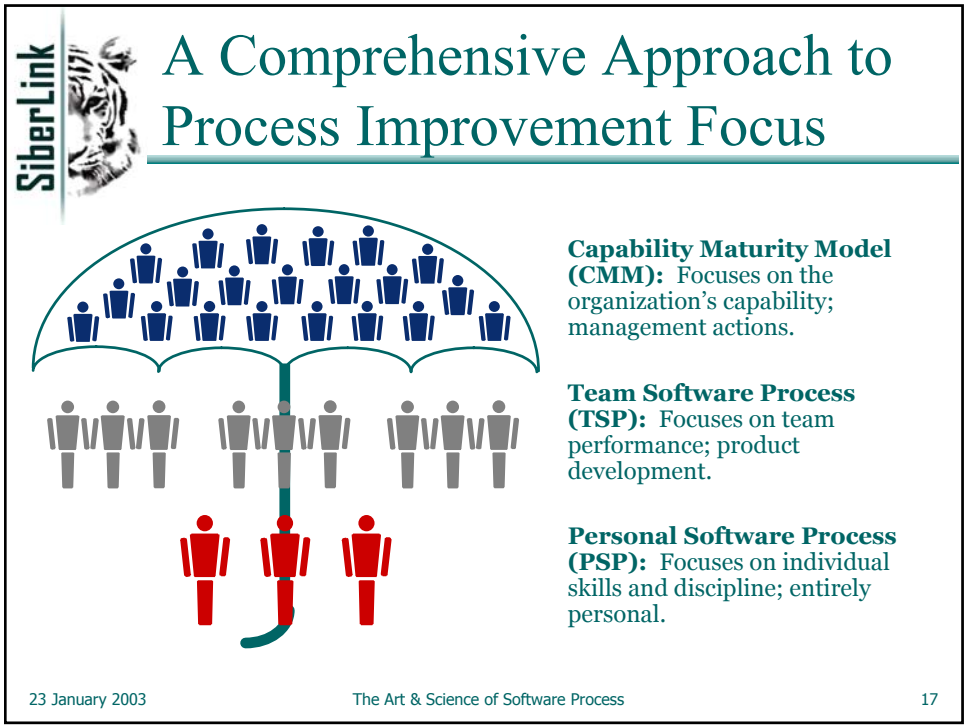
Armour, Phillip G. *The Five Orders of Ignorance*. Comm. of the ACM. Vol.43. No.10. Oct. 2000.

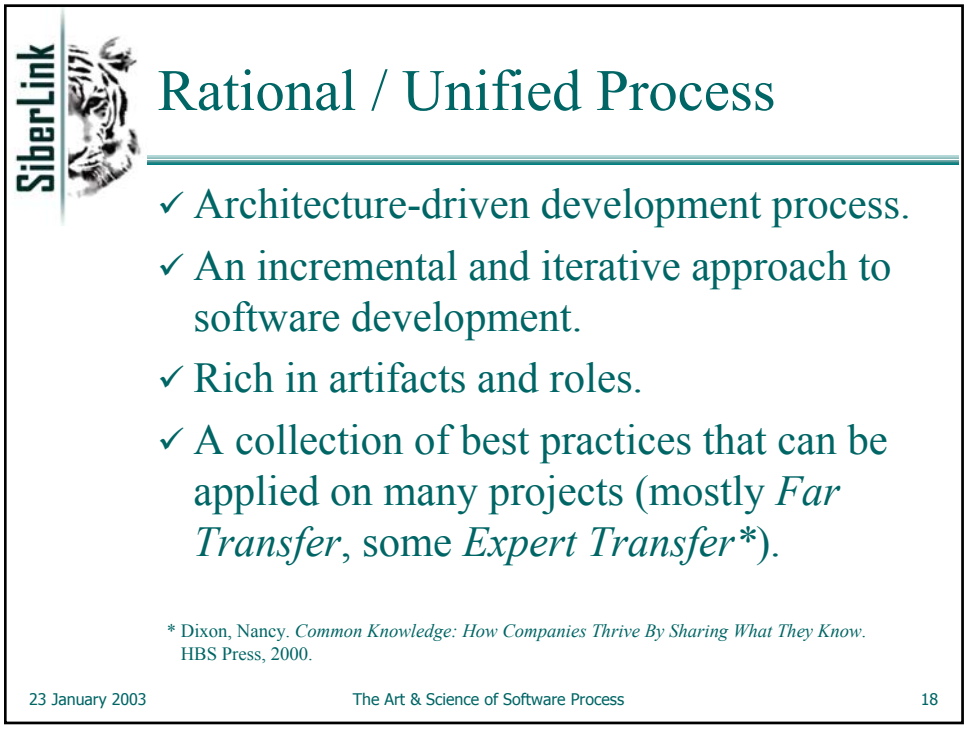

SiberLink
Good is the Enemy of Great!
"And that is one of the key reasons why we have so little that becomes great."
Collins, James C. Good to Great.Harper Business. New York, NY. 2001.
23 January 2003
The Art & Science of Software Process
13


SiberLink
Voyager—Total Defects in System Test
Voyager Spacecraft Defects in System Test
Cumulative Defects
0
20
40
60
80
100
120
140
1
9
17
25
33
41
49
57
65
73
81
89
97
105
Weeks
All
Non-Crit
Critical
23 January 2003


SiberLink
Approaches


SiberLink
Software Engineering Institute
Q: Who is the largest software consumer in the world?
A: The US Department of Defense.
Milestones
Mid ’80s: Capability Maturity Model (CMM) developed.
Early ’90s: Personal Software Process (PSP) is developed; class taught at Carnegie Mellon University.
Today: over 4,000 people trained in PSP Worldwide
Team Software Process: How can an organization create high performance software development teams.
23 January 2003
The Art & Science of Software Process
16


SiberLink
A Comprehensive Approach to Process Improvement Focus
Capability Maturity Model (CMM): Focuses on the organization's capability; management actions.
Team Software Process (TSP): Focuses on team performance; product development.
Personal Software Process (PSP): Focuses on individual skills and discipline; entirely personal.
23 January 2003
The Art & Science of Software Process
17

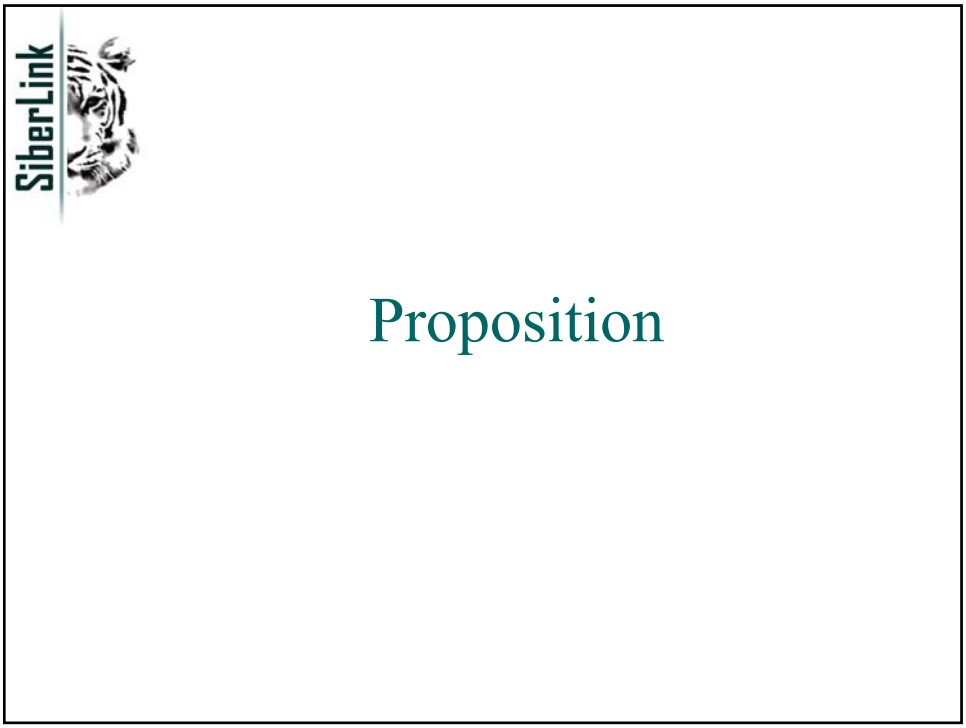

SiberLink
Rational / Unified Process
✓ Architecture-driven development process.
✓ An incremental and iterative approach to software development.
✓ Rich in artifacts and roles.
✓ A collection of best practices that can be applied on many projects (mostly Far Transfer, some Expert Transfer*).
* Dixon, Nancy. Common Knowledge: How Companies Thrive By Sharing What They Know. HBS Press, 2000.
23 January 2003
The Art & Science of Software Process
18

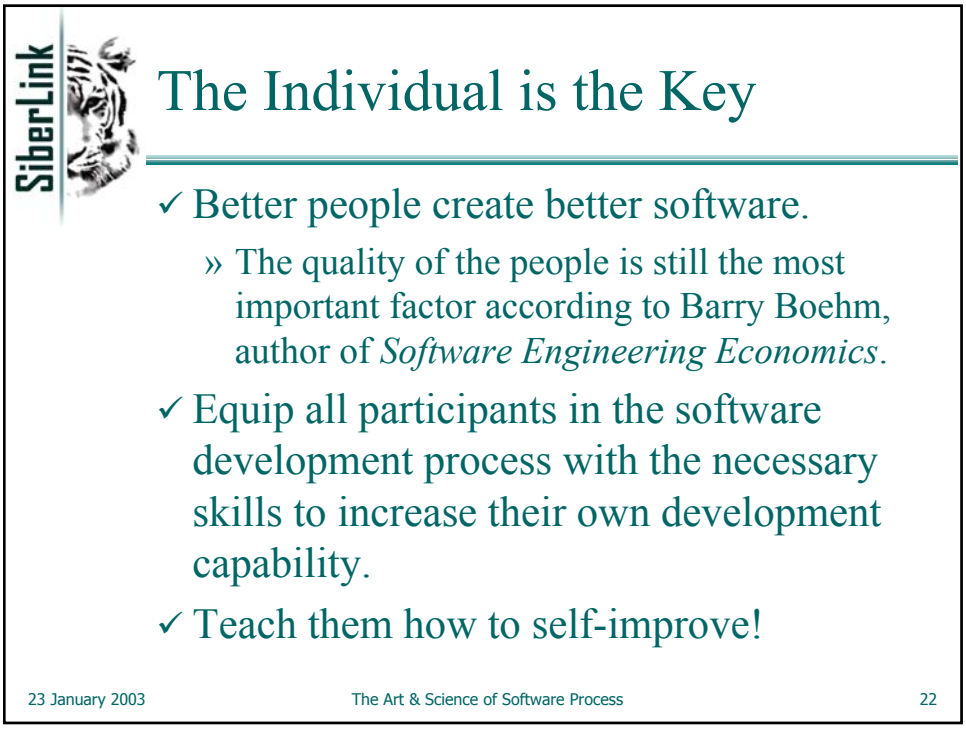
Extreme Programming

- ✓ Core practices:
  - » Whole Team
  - » Planning Game
  - » Small Releases
  - » Customer Tests
  - » Simple Design
  - » **Pair Programming**
  - » **Test-First Development**
  - » Design Improvement
  - » Continuous Integration
  - » Collective Code Ownership
  - » Coding Standard
  - » Metaphor
  - » Sustainable Pace
- ✓ *"Ruthlessly refactor."*

Other Approaches

- ✓ ISO 9001/9000-3
- ✓ SCRUM www.controlchaos.com
- ✓ FDD (Feature Driven Development)
- ✓ Agile Methodologies
- ✓ OPEN (Object-oriented Process, Environment, and Notation) www.open.org.au
- ✓ Code 'n Fix ☺


SiberLink
Proposition


SiberLink
The Individual is the Key
✓ Better people create better software.
» The quality of the people is still the most important factor according to Barry Boehm, author of *Software Engineering Economics*.
✓ Equip all participants in the software development process with the necessary skills to increase their own development capability.
✓ Teach them how to self-improve!
23 January 2003
The Art & Science of Software Process
22

Personal Mastery (Personal Process)

Senge, Peter. *The Fifth Discipline.* Currency Doubleday. New York, NY. 1990.

Personal (Software) Process

- ✓ Personal
  - » It is *your* process. If there is something that you don't like, then *you* need to change it!
- ✓ Software
  - » A personal process applied to software development.
- ✓ Process
  - » *"A series of actions, changes, or functions bringing about a result."*

Excerpted from *The American Heritage® Dictionary of the English Language*

Anybody who creates a deliverable that could have defects can benefit from a personal process.

Humphrey, Watts S. *A Discipline for Software Engineering.* Addison-Wesley. Reading, MA. 1994.

Why Focus on Yourself?

- ✓ You are distinct… or extinct.
- ✓ You are the same person at home, at work, at play.
- ✓ Think of yourself as **Me, Inc.**
  - » Even if you happen to be on somebody's payroll at the moment!

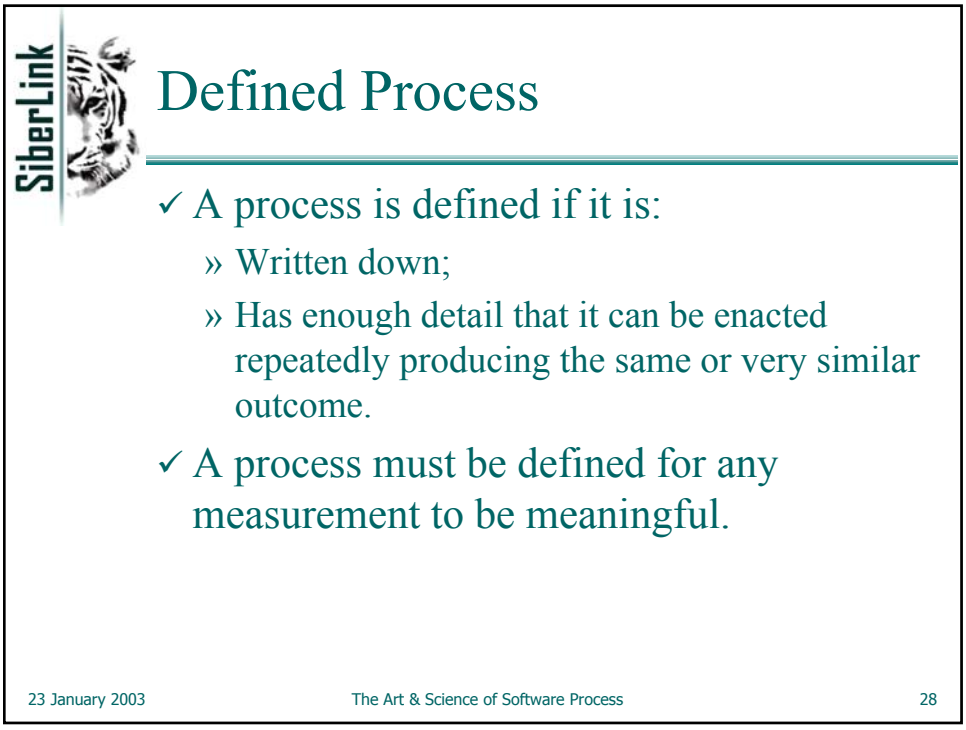
Peters, Thomas, J. *Brand You 50:Fifty Ways to Transform Yourself from an "Employee" into a Brand that Shouts Distinction, Commitment, and Passion*. Knopf/Random House, 1999.

Why Become Great?

- ✓ It is no harder to be great then to be mediocre. It takes **clarity & focus**.
- ✓ In the search of meaning when you find it, you will become great.
  - » For its own sake.

Collins, James C. *Good to Great.* Harper Business. New York, NY. 2001.


SiberLink
Elements of High-Performance
Software Development Practice


SiberLink
Defined Process
✓ A process is defined if it is:
» Written down;
» Has enough detail that it can be enacted repeatedly producing the same or very similar outcome.
✓ A process must be defined for any measurement to be meaningful.
23 January 2003
The Art & Science of Software Process
28

Planning

- ✓ Why? The plan is the basis of commitments. To be successful you must be able to make commitment that you can meet—**at a profit**.
- ✓ What is a plan? It is the amount of work that needs to be done to achieve the desired outcome.
- ✓ How? Plan in detail. Task length:45-90 minutes.
- ✓ Additional benefits:
  - » Identifies risks.
  - » Guides your work, enables you to be more efficient.
  - » Helps you track the status of the work.

Effective On-Task Time (EOT)

- ✓ The time effectively spent on the project.
- ✓ Doesn't include:
  - » Reading email (usually even if it is project related)
  - » Meetings (except well-defined project meetings)
  - » Lunch time, breaks, phone conversations, etc.
- ✓ Measure how many hours per week do you spend doing project work, that's your EOT per week.
  - » Best organizations in the world get 20+ hrs/week.
  - » You may only get about 3-5 hrs/wk the first week. You should get up to 15 hrs/wk in a couple of weeks.

Research vs. Development

- ✓ Research:
  - » Inventing something new, that has never existed.
  - » It can only be time limited. When the time is up, make a decision: continue, or seek an alternative solution.
- ✓ Development:
  - » Use existing technology, or implement an invention.
  - » Can be planned & scheduled; it has been done before.
- ✓ *Library research* and *learning* can be scheduled.

Context

- ✓ What is context?
  - » Everything that is said, done, drawn, or written during the software development process.
- ✓ How much context do you need?
  - » Just enough to always know where you are with the work and to know what to do next.

Component-Based Development

- ✓ Decompose the problem into a set of cooperating components.
- ✓ Assemble your software from high-quality components.
- ✓ If you can write high-quality components you have a chance of creating high-quality large programs.

Estimation

- ✓ Size (e.g. KLOC for code): estimate only this!
  - » Calculate time, schedule, & defects based on size.
- ✓ Time (project hours)
  - » Calculate time based on historical productivity data. If productivity data is not available then estimate it.
  - » Work in 1-2 week iterations. Use the current productivity data to adjust the plan.
- ✓ Schedule (map project hours to calendar days)
  - » Schedule is the hours available for project work.
- ✓ Defects (e.g. Defects / KLOC)
  - » Estimate defects based on size using historical defect injection data.

Quality Planning

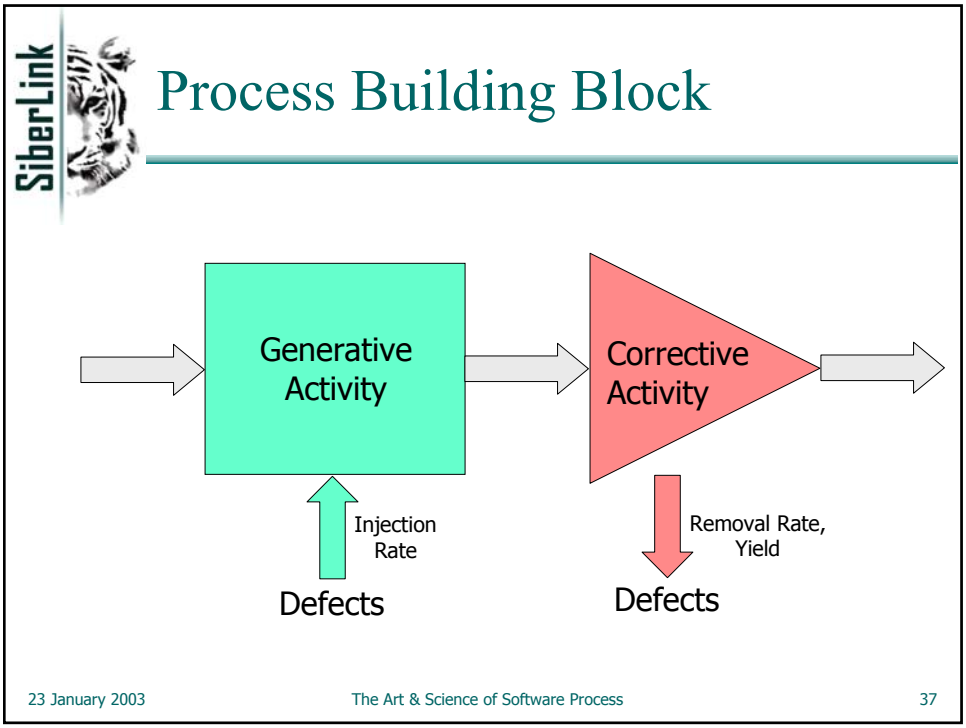
- ✓ Must change your process to change your results.
  - » *Doing the same thing over and over and expecting a different result = Insanity!*
- ✓ You know that you will put the defects in, might as well plan to remove them.
- ✓ Use your defect injection rate per phase to calculate how many defects you have to remove and plan the removal activities accordingly.
- ✓ Some removal activities are more efficient then others, get the data to see where you get the most bang for the buck.

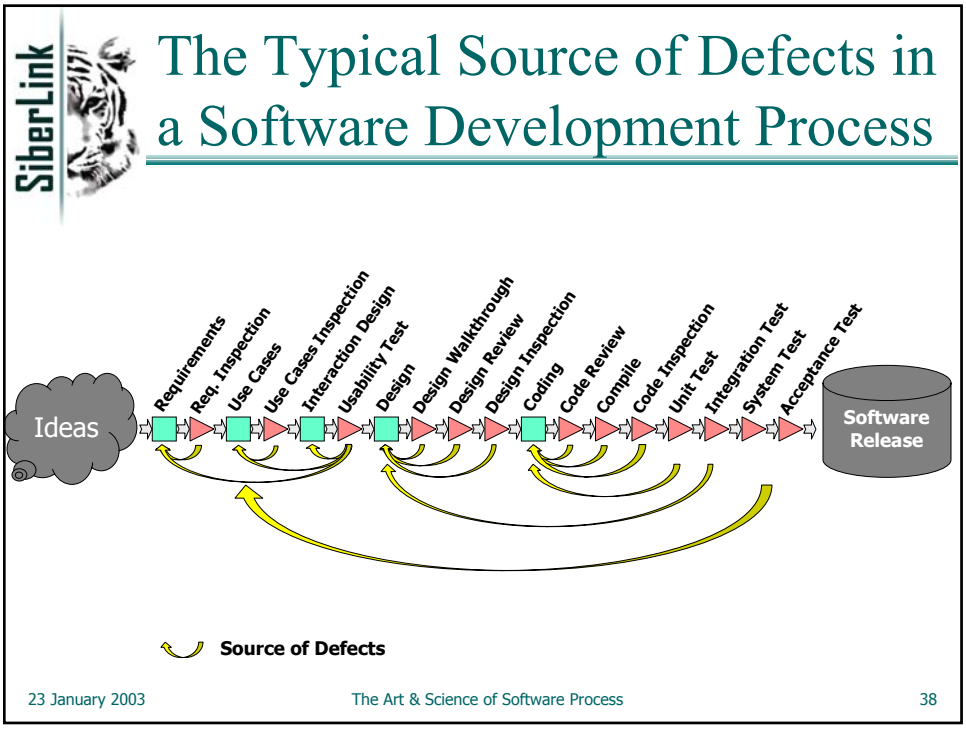
Broken Windows & Software?

- ✓ The brainchild of criminologist James Q. Wilson and George Kelling.
- ✓ Crime is the inevitable result of disorder.
- ✓ If one is broken, soon more will be.
- ✓ Applies to software equally well…

❑ Not Microsoft Windows ☺

Gladwell, Malcolm. *The Tipping Point: How Little Things Can Make A Big Difference*. Pg. 141. Little, Brown, and Company. New York, NY 2000.


SiberLink
Process Building Block
Generative Activity
Corrective Activity
Injection Rate
Defects
Removal Rate, Yield
Defects
23 January 2003
The Art & Science of Software Process
37


SiberLink
The Typical Source of Defects in a Software Development Process
Ideas
Requirements
Req. Inspection
Use Cases
Use Cases Inspection
Interaction Design
Usability Test
Design
Design Walkthrough
Design Review
Design Inspection
Coding
Code Review
Compile
Code Inspection
Unit Test
Integration Test
System Test
Acceptance Test
Software Release
Source of Defects
23 January 2003
The Art & Science of Software Process
38

Ongoing Process Improvement

- ✓ You need to **know** what your process is so you can improve it!
- ✓ In workplaces where people **understand** the process and follow it, they write several improvement proposals per week.
- ✓ Write a Process Improvement Proposal (PIP) for yourself as soon as you think of some improvement and periodically review them and incorporate some or all into your work.
- ✓ *Improvement isn't possible if your process doesn't change; "working hard" doesn't cut it.*

Data Analysis

- ✓ Collect data for a reason! If you never look at the data you collected, then don't collect it!
- ✓ Data can tell you:
  - » Where your time goes? What did you really work on?
  - » What was forgotten from the plan? What was extra?
  - » Where can you improve? … and many more things!
- ✓ Watch out! It can be a mirror that might not be pleasant to look at, but don't be discouraged, everybody has areas for improvement.
- ✓ **The data belongs to you!** You decide who you show it to. You collect data for your own benefit.

This Works!

AIS Schedule Data

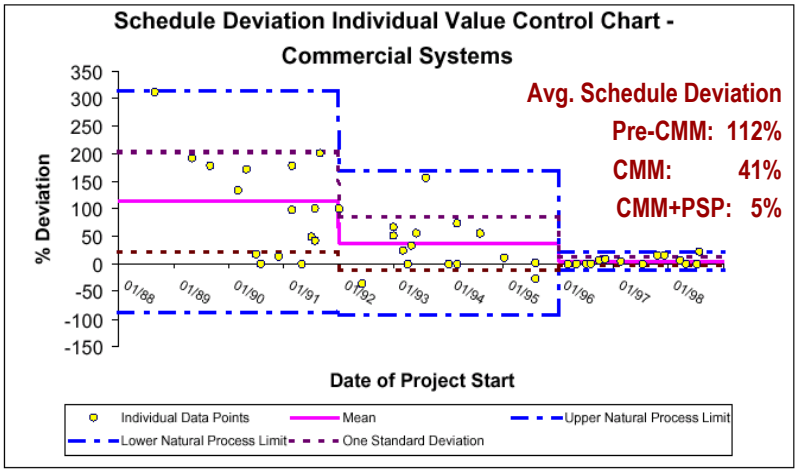


Figure 10: Schedule Deviation Individual Value Control Chart - Commercial Systems

23 January 2003

Conclusions

Is it *Art & Science*?

- ✓ It is and will be.
- ✓ As we understand more about:
  - » software development,
  - » people,
  - » processes,
  - » relationship to other domains

…we will evolve part of the *Art* into *Science* and we will continually discover new *Art* that we need to master.

Can A Personal Process Help?

- ✓ Less defects in your work.
- ✓ Better understanding of what is needed to complete a project. You can tell management or the client when you need more information.
- ✓ Better estimation skills so ***you* can *make commitments that you can keep***. In turn the business can make commitments as well.
- ✓ Better project tracking skills. Increased visibility into the project status.
- ✓ Caveat: Your productivity will drop in the short term. You are learning a new way to work. It takes time to became expert in a new skill.

Your Theories Lead You

*The way you work depends on your **thinking**!*

- ✓ You live with your personal (software) process.
- ✓ Getting another psp then the one you have, means you have to change the way you think and work.
- ✓ It is up to you to work in the most productive way for you!
- ✓ For your own sake you should know your performance!
- ✓ It is possible to create defect free code.

Closing Quote

"If things seem under control,
you are just not going fast enough."

—Mario Andretti, race-car driver

Thank You!

✓ Contact Information

Steven Teleki
Vice President, Software Engineering
SiberLink, Inc., http://siberlink.com/
2720 Bee Caves Road
Austin TX 78746

teleki@siberlink.com or **teleki@computer.org**

For a software development reading list please visit:
http://pseng.net/