



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.
18 September 2002

The Art & Science of Software Process

2




The Goal of Software Process




The Goal of Software Process

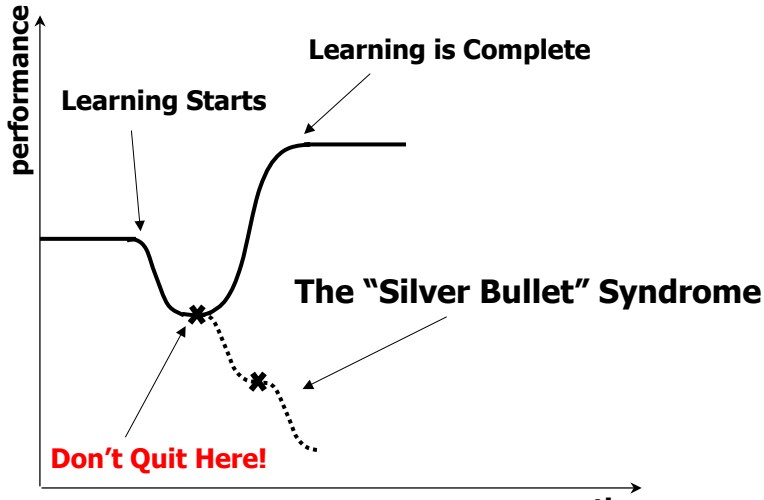
- ✓ Make commitments that you can keep.
- ✓ *Produce quality software on-time and on-budget.*

SiberLink 

Challenges

SiberLink 

Learning Takes Time!



performance

Learning Starts

Learning is Complete

The "Silver Bullet" Syndrome


Don't Quit Here!

time

18 September 2002

The Art & Science of Software Process

6


SiberLink 

You'll Need Common Sense!

Documented Processes

		Yes	No
Common Sense	Yes	Quality	Creative Chaos
	No	Mindless Bureaucracy	Mindless Chaos

From Mark Paulk, with thanks to Sanjiv Ahuja, President and COO of Bellcore.
18 September 2002 The Art & Science of Software Process 7

SiberLink 

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.

18 September 2002 The Art & Science of Software Process 8



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.
18 September 2002

The Art & Science of Software Process

9



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.
18 September 2002

The Art & Science of Software Process

10



Good is the Enemy of Great!

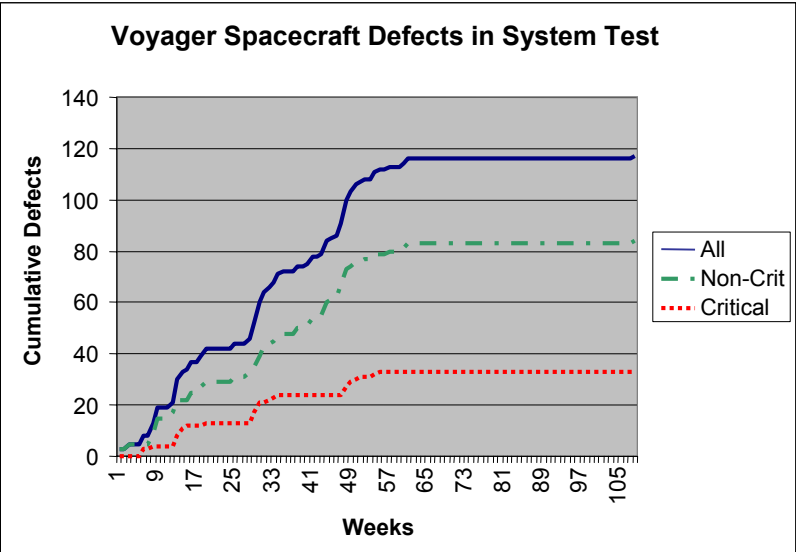
“And that is one of the key reasons why we have so little that becomes great.”

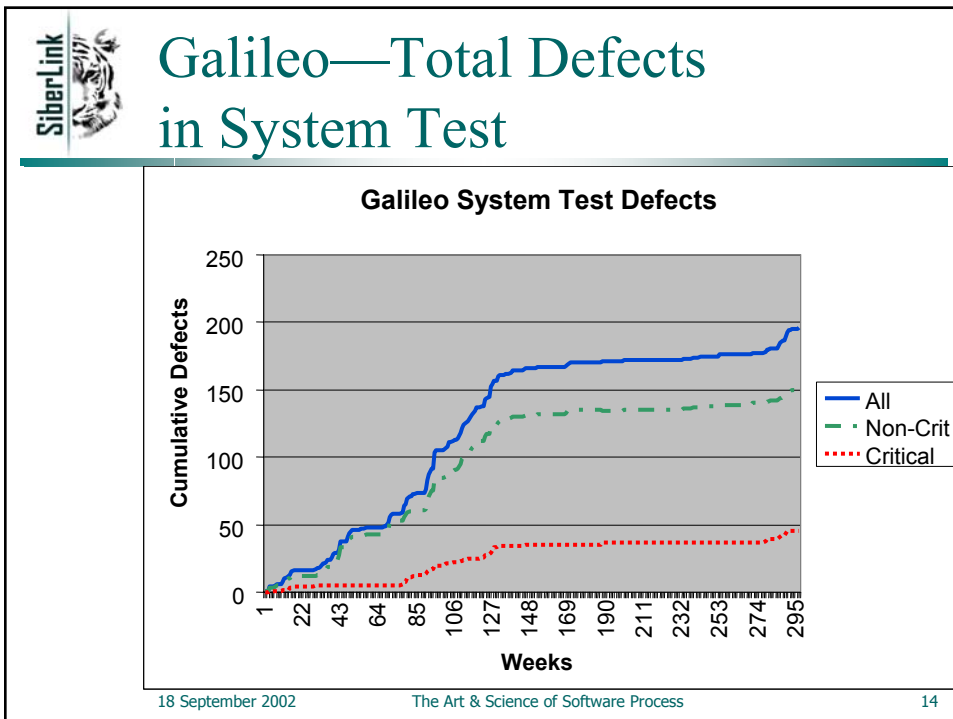
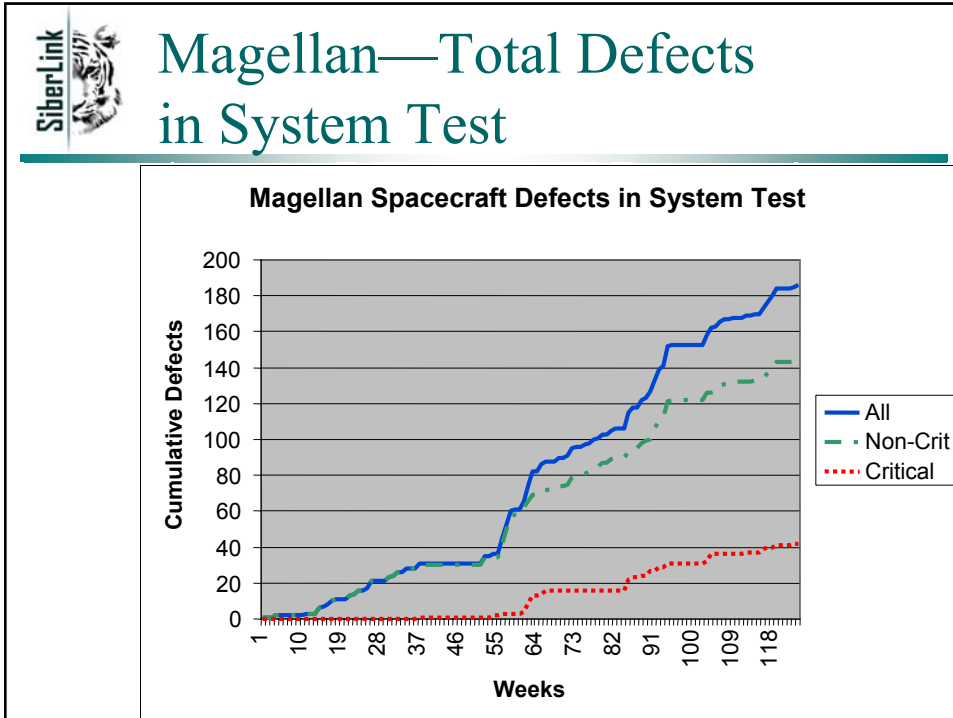
Collins, James C. *Good to Great*. Harper Business. New York, NY. 2001.
 18 September 2002 The Art & Science of Software Process

11



Voyager—Total Defects in System Test





18 September 2002

The Art & Science of Software Process

14



Approaches




Software Engineering Institute

Q: Who is the largest software consumer in the world?

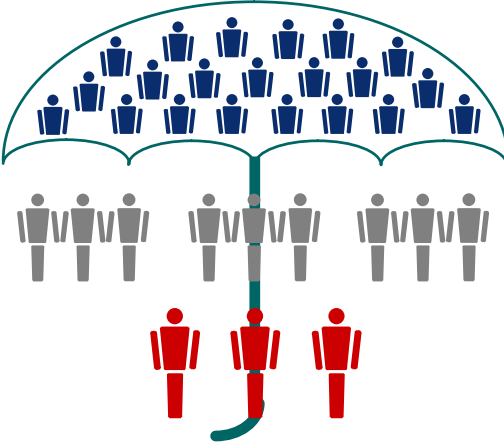
A: The US Department of Defense.

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.



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.

18 September 2002
The Art & Science of Software Process
17



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.

18 September 2002
The Art & Science of Software Process
19



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.”*

18 September 2002

The Art & Science of Software Process

20



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 ☺

18 September 2002

The Art & Science of Software Process

21



Proposition



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 improve their own development skills.
- ✓ Teach them how to self-improve!



Personal Mastery (Personal Process)



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 does anything that involves creating a deliverable that could have errors can benefit from a personal process.



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.

18 September 2002

The Art & Science of Software Process

26



Why Become Great?

- ✓ It is no harder to be great than 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.

18 September 2002

The Art & Science of Software Process

27



Elements of High-Performance Software Development Practice



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 possible.
- ✓ If the process is not defined, the measurement is meaningless.



Planning

- ✓ Why? Because 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? The plan represents the amount of work that needs to be done to achieve the desired outcome.
- ✓ How? Plan in detail; task length about 45 to 90 minutes.
- ✓ Other benefits:
 - » Identifies risks.
 - » Guides your work, enables you to be more efficient.
 - » Helps you track the status of the work.

18 September 2002

The Art & Science of Software Process

30



Effective On-Task Time (EOT)

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

18 September 2002

The Art & Science of Software Process

31



Research vs. Development

- ✓ Research:
 - » You have to invent something new, that has never existed.
 - » It can only be time limited. When the time is up, evaluate the situation and make a decision: decide whether to continue, or to seek an alternative solution.
- ✓ Development:
 - » You have to use existing technology, or implement a new invention.
 - » Can be planned & scheduled since it has been done before.
- ✓ If you are doing “*library research*” then say it so. This can be scheduled.

18 September 2002

The Art & Science of Software Process

32



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 development and what to do next.

18 September 2002

The Art & Science of Software Process

33



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, then you have a chance of creating high-quality large programs.

18 September 2002

The Art & Science of Software Process

34



Estimation

- ✓ Size (e.g. KLOC for code)
 - » Estimate size only.
 - » 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. At the end of each iteration you have current productivity data. Adjust the plan accordingly.
- ✓ Schedule (map project hours to calendar days)
 - » Schedule is the number of hours available for project work.
- ✓ Defects (e.g. Defects / KLOC)
 - » Estimate defects based on the size using historical defect data.

18 September 2002

The Art & Science of Software Process

35



Quality Planning

- ✓ As long as you don't change your process your results will be the same and the new product will contain about the same amount of defects that the old one had.
- ✓ If you know that you will put the defects in, might as well plan on trying to remove them.
- ✓ If you know your historical injection rate per phase, then you can figure out how many defects you will have to remove and plan removal activities.
- ✓ Some removal activities are more efficient than others, you got to get the data to figure out where do you get the most bang for the buck.

18 September 2002

The Art & Science of Software Process

36



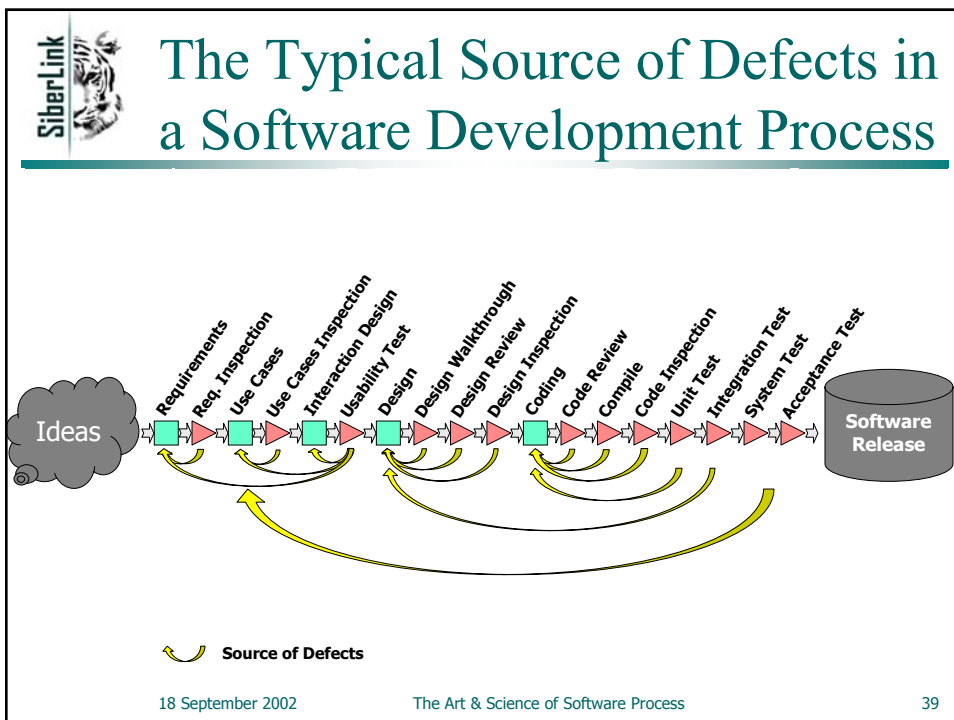
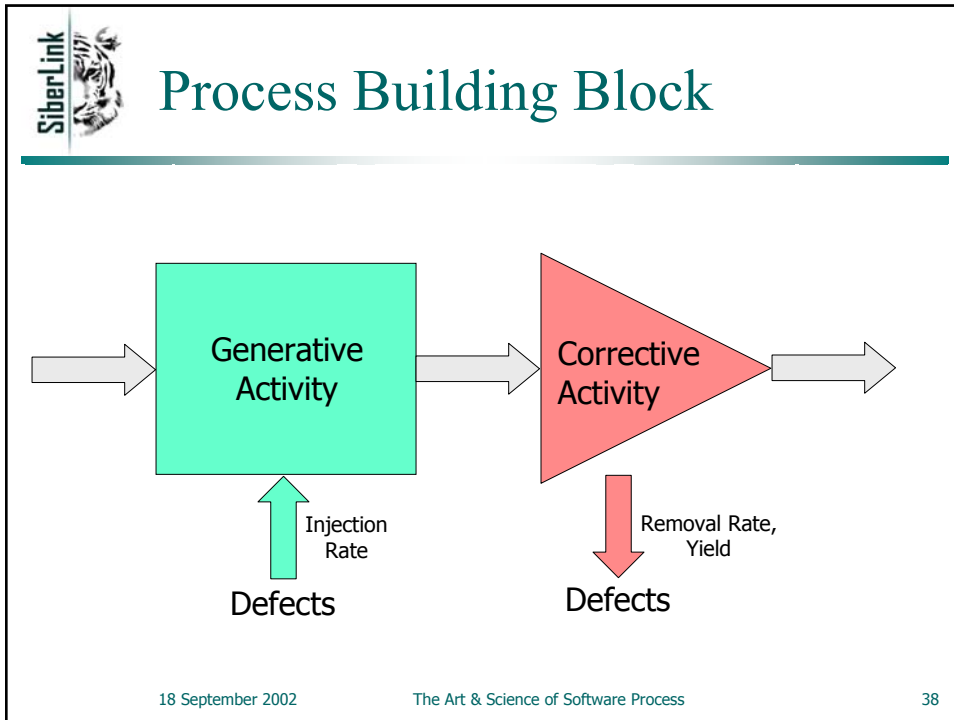
Simple ROI Calculation

Current	Averages		Example	
			20,000	LOC
	5	defects/KLOC	100	defects
	10	hours/defect	1,000	hours
	125	\$/hour	125,000	\$
	10	LOC/hr	2,000	hours
	20	hours/week	100	weeks/dev
	4	developers	25	weeks/team
Future	Averages		Example	
			20,000	LOC
	1	defects/KLOC	20	defects
	10	hours/defect	200	hours
	125	\$/hour	25,000	\$
	10	LOC/hr	2,000	hours
	20	hours/week	100	weeks/dev
	4	developers	25	weeks/team
Savings			800	hours
			100,000	\$

18 September 2002

The Art & Science of Software Process

37





Ongoing Process Improvement

- ✓ You want 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.*

18 September 2002

The Art & Science of Software Process

41




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.

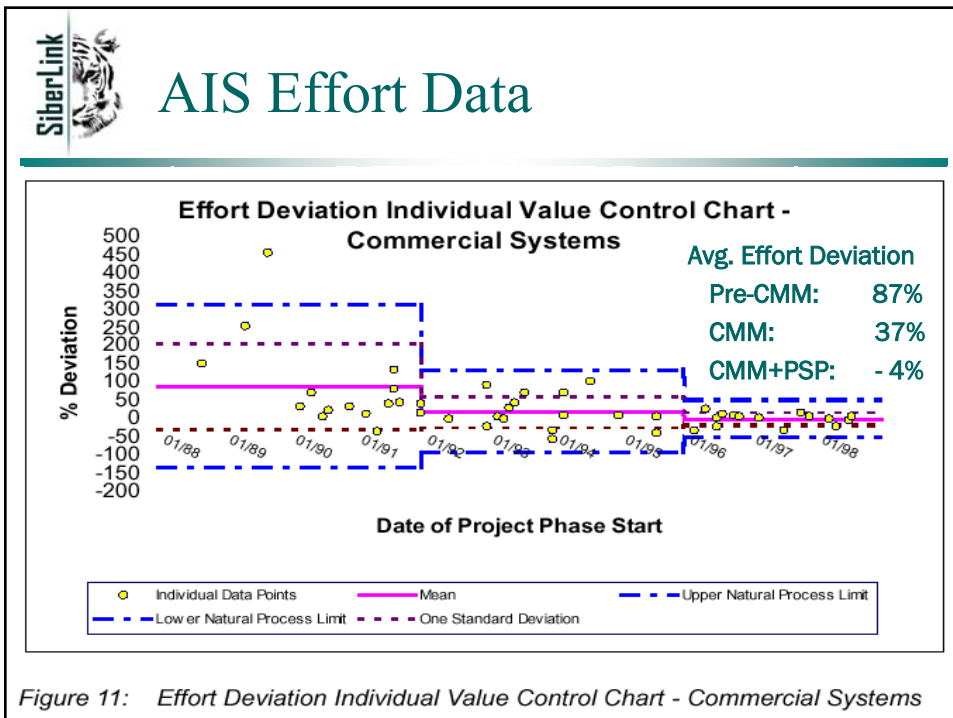
18 September 2002

The Art & Science of Software Process

42



This Works!





Conclusion



What Should You Expect From A Disciplined Personal Process?

- ✓ 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 to finish the work.
- ✓ Better estimation skills so ***you can make commitments that you can keep***. In turn the business can make—and keep—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 become expert in a new skill.



Summary

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

- ✓ You live with your own personal software process (psp).
- ✓ Getting a different psp than 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 write defect free code.

18 September 2002

The Art & Science of Software Process

54



Summary (continued)

- ✓ A defined process will enable you to understand, monitor, and improve your performance.
- ✓ As you work, record accurate and complete data on your process.
- ✓ Use the collected data to improve what you do!

18 September 2002

The Art & Science of Software Process

55



Closing Quote

*“If things seem under control,
you are just not going fast enough.”*

—Mario Andretti, race-car driver

18 September 2002

The Art & Science of Software Process

56



Thank You!

✓ Contact Information

Steven Teleki

Vice President, Software Engineering

SiberLink, Inc., <http://www.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/>

18 September 2002

The Art & Science of Software Process

57