Carnegie Mellon
Software Engineering Institute

Scaling Up the Process

Watts S. Humphrey

The Software Engineering Institute

Carnegie Mellon University

Carnegie Mellon
Software Engineering Institute

Agenda

Development Performance

Future Challenges

Current Problems

Scalability

Process Principles and Requirements

A Superior Process

Carnegie Mellon
Software Engineering Institute

Performance Examples

The IRS system – finally started to use in 2005

- 5 years of delays
- costs exploded to $2 B

FBI system killed

- 3 years of work
- $150 M
- 5 CIOs, 9 program managers

Britain's child-support project

- a year late
- $844 M
- didn't pay 50% of cases

Carnegie Mellon
Software Engineering Institute

Transportation

Automobiles

- Mercedes Benz – batteries, windows, temperature
- Bendix Brakes
- Ford Explorer

Boeing 777

- Malaysian Airlines
- Singapore

Carnegie Mellon
Software Engineering Institute

This Is No Joke!

These are not new problems.

- CONFIRM system 20 years ago
- Cancelled after 3 ½ years and $125 M

How long will society tolerate such performance?

Do we want government controls?

- Methods standards and approval?
- Pre-shipment reviews?
- Legislated warranties?

We had better solve our own problems or others will solve them for us.

Carnegie Mellon
Software Engineering Institute

Sir Francis Bacon

"He who will not apply new remedies must expect new evils."

Carnegie Mellon
Software Engineering Institute

Challenges of the Future

Our priorities must change.
Systems are now

- larger
- distributed
- integrated
- pervasive
- critical

The methods of the past are not suitable today.

In the future, they could be dangerous!

Carnegie Mellon
Software Engineering Institute

Systems Development Phases

Phase I – Feasibility
- after World War II
- mid 1960s

Phase II – Manageability
- early to mid 1960s
- feasibility - lower priority
- still continuing

Phase III – Quality
- now
- manageability - lower priority
- priorities: safety, security, privacy, usability, etc.

Carnegie Mellon
Software Engineering Institute

Process Criteria

A superior process must meet three criteria.

- work effectively for the smallest and largest programs
- consistently produce superior results
- be recognized as producing superior results

The current generally-used processes

- do not consistently work for small projects
- rarely work effectively for large projects
- produce inconsistent and often poor-quality results

Carnegie Mellon
Software Engineering Institute

Superior Results

A superior process must consistently deliver products on schedule and for their committed costs.

It must routinely deliver high-quality products.

- functions
- properties
- defects

It must dynamically respond to changing needs.

Carnegie Mellon
Software Engineering Institute

Systems Properties – 1

Systems have emergent properties.

- Emergent properties are produced by synergies among the system's components.
- No single component supplies any emergent property.

Example emergent properties

- performance
- reliability
- safety
- security
- usability
- maintainability
- installability

Carnegie Mellon
Software Engineering Institute

Systems Properties – 2

The quality of an emergent property is determined by the

- worst-quality component
- lowest-performing component
- least-reliable component
- least usable component
- least secure component

Therefore, a superior process must

- identify all poor-quality components
- fix all poor-quality components
- improve all poor-quality components
- prevent all poor-quality work

Carnegie Mellon
Software Engineering Institute

The Scale-Up Problem

As systems become larger

- they become more complex
- their development is more challenging
- their management is more difficult
- new problems emerge at the systems level
- component problems are magnified at higher levels

Therefore, a superior process must

- prevent all poor-quality work
- guide the work at all levels

Tolstoy: Anna Karenina

"Happy families are all alike; every unhappy family is unhappy in its own way."

Carnegie Mellon
Software Engineering Institute

The Common Strategy

The most common strategy:

- If it isn't broken – don't fix it.

This strategy has four problems.

- Processes break in an infinite number of ways.
- Fixing one problem will not fix the next.
- These random fixes do not fix causes.
- The process is not consistently or measurably improved.

Carnegie Mellon
Software Engineering Institute

The CMMI Strategy

The only effective answer is a strategy that is

- cohesive
- comprehensive
- based on sound principles

This is the logic for CMMI.

It has gaps.

Scalability

Scalability is fundamental.

Developers now typically use the same practices for

- a 1,000 LOC application
- a 1,000 LOC module for a 1,000,000 system

Examples from other fields.

- boat building
- building construction
- accounting

Carnegie Mellon
Software Engineering Institute

Scalability Requirements

For a process to be scalable, every key aspect must scale up and down.

- practices
- measures
- quality management

Therefore, the work at every level must

- be based on precise and detailed plans
- produce high-quality products
- be based on measured, statistically usable, and auditable data, not
  - after-the-fact reports
  - third-party estimates
  - guesses

Carnegie Mellon
Software Engineering Institute

Meeting Schedules

Fred Brooks: “Schedules slip a day at a time.”

Carnegie Mellon
Software Engineering Institute

A Process Principle

To consistently maintain costs and schedules, the one-day slips must be recovered the next day. If not

- the delays will compound
- the commitments will become unmanageable

With current common practices, schedule slips cannot be detected until projects are weeks or months late.

Then it is generally too late to recover.

Carnegie Mellon
Software Engineering Institute

System Schedules

For large-scale integrated-systems-development programs

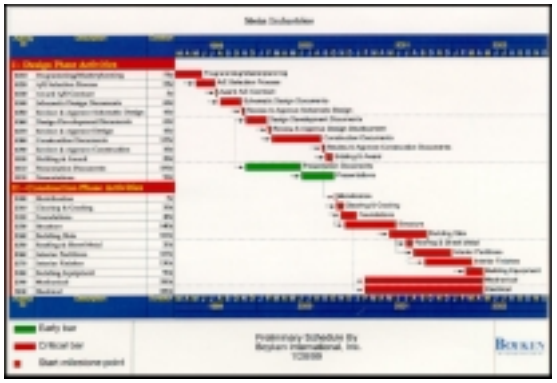
- Any subsystem delay will delay the system.
- Any component delay will delay the subsystem and system.
- Any project delay will delay the component, subsystem, and system.
- Any team delay will delay the project, component, subsystem, and system.
- Any developer delay will delay the team, project, component, subsystem, and system.

Carnegie Mellon
Software Engineering Institute

Program Management

To manage large programs, every system level must be managed.

- subsystems
- components
- projects
- teams
- developers



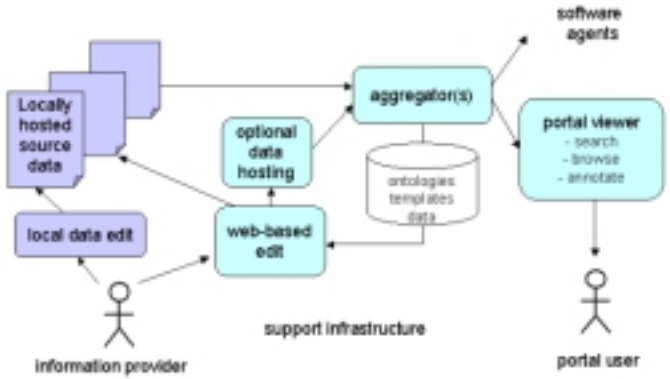
Therefore, to consistently produce quality products on schedule and for planned costs, every developer must

- consistently produce quality products
- predictably meet schedules

Process Requirements

For a process to meet these needs, it must

- properly manage the knowledge work
- manage costs and schedules
- manage quality



Carnegie Mellon
Software Engineering Institute

Knowledge Work - 1

The first rule for managing knowledge work is to recognize that you can't manage it.

The knowledge workers must manage their own work.

The second rule for managing knowledge work is that the teams and developers must

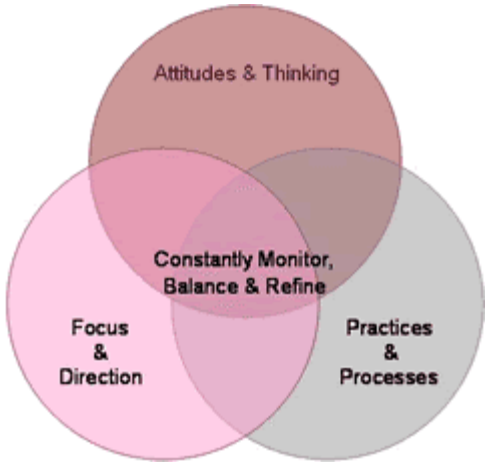
- know how to manage themselves
- negotiate their commitments with management
- manage with data
- own their own work

Carnegie Mellon
Software Engineering Institute

Knowledge Work - 2

To manage themselves, developers must

- use personal processes
- follow personal plans
- measure, track, and report status
- measure and manage quality



Carnegie Mellon
Software Engineering Institute

Knowledge Work - 3

To follow a defined, planned, measured, and quality-controlled process, developers must

- work on self-directed teams
- define their own processes and plans
- negotiate their own commitments
- be competently led and coached

Carnegie Mellon
Software Engineering Institute

Cost and Schedule

There is no secret to meeting cost and schedule commitments.

In every field, only four things are required.

- The developers estimate and plan their own work.
- Everyone precisely and regularly tracks and reports status and progress.
- Schedule delays are addressed every day.
- When requirements change, everyone
  - revises their plans
  - renegotiates their commitments

Carnegie Mellon
Software Engineering Institute

Quality Management - 1

To successfully manage quality, the development process must be based on four facts.

- The longer a defect remains in a product, the more it costs to find and fix it.
- No test can find more than a fraction of the defects in a product.
- The larger and more complex the product, the smaller this faction will be.
- To get a high-quality product out of test, you must put a high-quality product into test.

Carnegie Mellon
Software Engineering Institute

Quality Management - 2

A quality process must strive to find and fix all defects before test entry.

To do this, the process must

- have multiple early defect-removal steps
- measure product quality at every step
- measure process quality at every step
- make and follow personal and team quality plans
- regularly track product and process quality
- use testing to
  - verify product quality
  - gather quality data
- promptly correct quality deviations from plan

Carnegie Mellon
Software Engineering Institute

A Superior Process

The TSP helps organizations to

- control costs
- improve product quality
- reduce cycle time

Software developers like the TSP because it

- shows them how to manage their own work
- improves their productivity
- enables them to consistently meet commitments
- improves their quality of life

Carnegie Mellon
Software Engineering Institute

Process Results

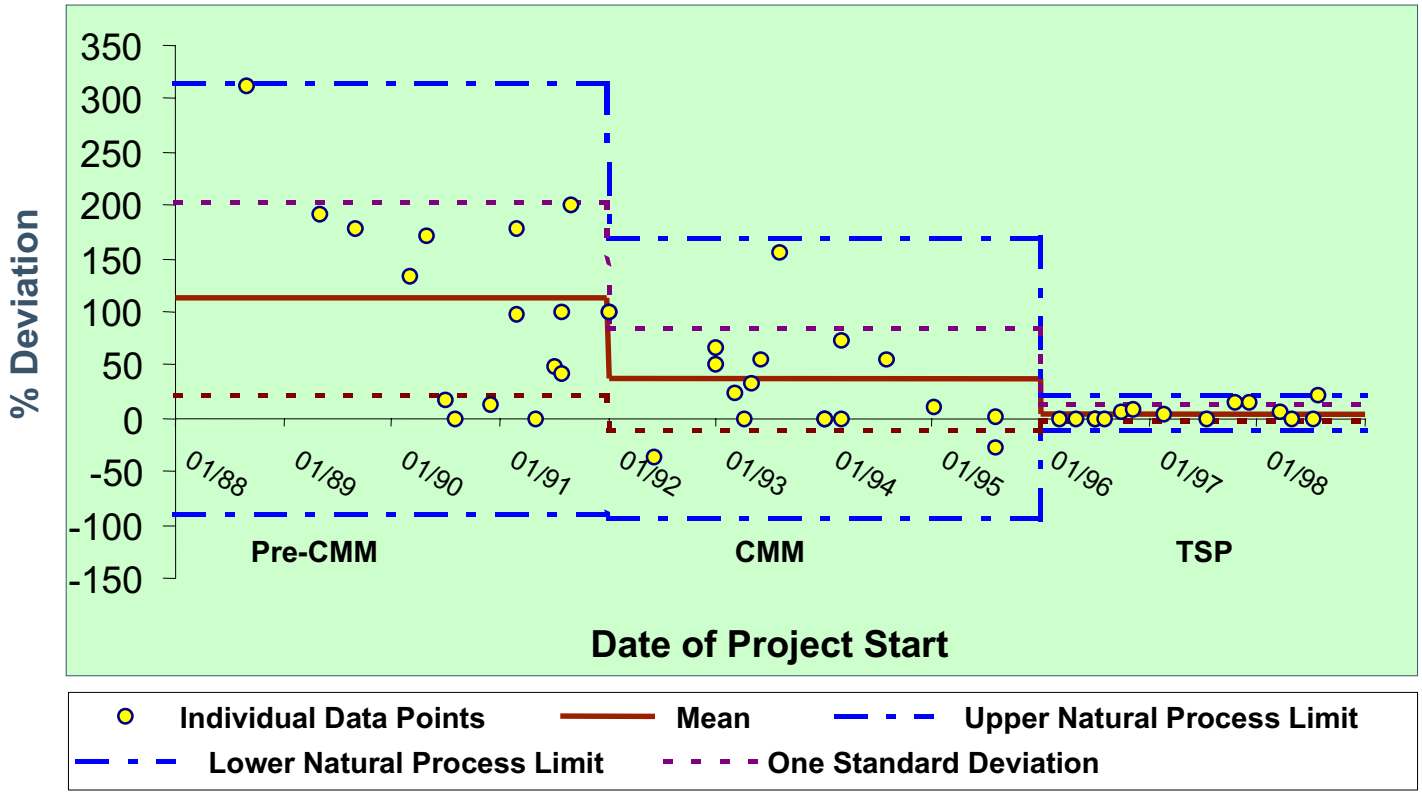
Some of the best known organizations introducing the TSP are

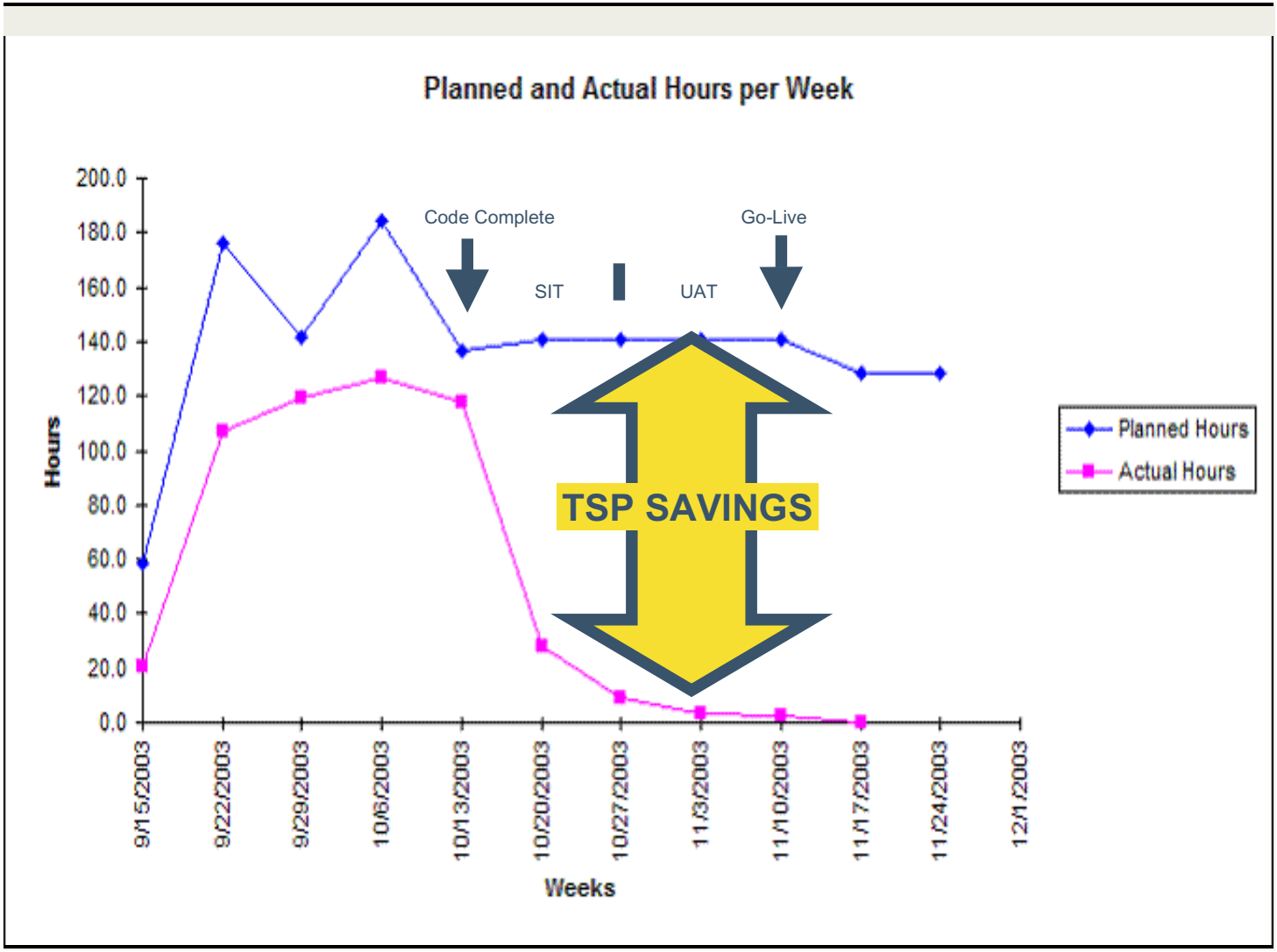
ABB	Lockheed
AIS	Microsoft
Bechtel-Bettis	NASA Langley
Boeing	NCR Teradata
DFAS	Northrop Grumman
EDS-SDRC	Oracle
Erickson	Teradyne
Honeywell	USAF: Hill AFB
Intuit	USN: NAVAIR
Kaiser	Xerox

Carnegie Mellon
Software Engineering Institute

The AIS Corporation

Schedule Deviation Control Chart




Planned and Actual Hours per Week
Code Complete
Go-Live
SIT
UAT
TSP SAVINGS
Planned Hours
Actual Hours
Hours
Weeks
200.0
180.0
160.0
140.0
120.0
100.0
80.0
60.0
40.0
20.0
0.0
9/15/2003
9/22/2003
9/29/2003
10/6/2003
10/13/2003
10/20/2003
10/27/2003
11/3/2003
11/10/2003
11/17/2003
11/24/2003
12/1/2003

Team Development

Team performance improves with experience. In 4 releases, one team

- delivered on time
- increased productivity by 81%

Defect Data - Five Releases of the Same Product					
	Non-TSP	TSP	TSP	TSP	TSP
Release number	1	2	3	4	5
Defects/1000 LOC					
System Test	13.3	2.5	0.8	1.0	0.1
Acceptance Test	0.8	0.0	0.0	0.0	0.0
Customer Production	1.2	0.2	0.0	0.0	0.0

Carnegie Mellon
Software Engineering Institute

An Industrial Study

A comparison of the performance of 80 traditional development teams with 15 first-time TSP teams

	Non-TSP Projects	TSP Projects
Released on Time	42%	66%
Average Days Late	25	6
Mean Schedule Error	10%	1%
Test Defects/KLOC	19.5	12.8
Production Defects/KLOC	1.8	0.5
Sample Size	80	15

Carnegie Mellon
Software Engineering Institute

Conclusions

These principles have been proven with CMMI and TSP and are being piloted for systems engineering with TSPI.

The critical challenge is to get people to work this way.

Consistently superior knowledge work must include the following elements – at all levels.

- well-defined and consistently practiced principles of personal performance
- teambuilding and empowerment
  - ownership and commitment
  - leadership and coaching
- organizational support and institutionalization
- quantitative performance measurement and appraisal

Carnegie Mellon
Software Engineering Institute

For More Information

Visit the PSP/TSP Web site

http://www.sei.cmu.edu/tsp/

Contact an SEI partner

http://www.sei.cmu.edu/collaborating/partners/trans.part.psp.html

Contact SEI customer relations

Phone, voice mail, and on-demand FAX: 412/268-5800

E-mail: customer-relations@sei.cmu.edu

See the books

***Winning with Software*, by Watts Humphrey, Addison-Wesley, 2002**

***TSP: Leading a Development Team*, by Watts Humphrey, Addison-Wesley, 2006**

***TSP: Coaching Development Teams*, by Watts Humphrey, Addison-Wesley, 2006**