Software Engineering Process Improvement

Self-Directed Teams and Data-Driven Management

Dr. Mark Sebern, PE
414-277-7213
sebern@msoe.edu
people.msoe.edu/~sebern/

Before we begin, it may be helpful to define some terms.

SEPI
- Software Engineering Process Improvement
  - Training courses
  - Coaching and implementation support

PSP℠
- Personal Software Process℠

TSP℠
- Team Software Process℠

Watts Humphrey and the TSP team
Software Engineering Institute (SEI)

BEC
- Business Excellence Consortium [bec.msoe.edu]
  - Milwaukee School of Engineering (MSOE)
  - Industry training and implementation support

While their pioneering work is gratefully acknowledged, neither Mr. Humphrey nor the SEI are responsible for the SEPI materials, and no claim is made that these materials represent any software process with which they are associated.

Please allow me to introduce myself …

Professor, Milwaukee School of Engineering (1994-)
- Founding program director for the undergraduate software engineering program (BSSE)
  - One of the first accredited SE programs in USA
  - Initiated 1999, accredited 2002-2003

Visiting Scientist, Software Engineering Institute
- Sabbatical leave 2004-2005
- Not now a representative of, or affiliated with, the SEI

Industry experience (20 years, prior to MSOE)
- Digital Equipment Corporation
- Sebern Engineering Inc. (17 years)

Over the last half century or so, software has become pervasive throughout industry, society, government, and the global economy.

IT professionals and software engineers are at the heart of this growth.

This widespread application of software products and systems has, however, created a new set of challenges.

The complexity and scale of the software systems we must build and maintain has grown exponentially.

Mission-critical systems demand very high levels of quality and reliability.

Limited resources and shortened schedules place a premium on predictability and productivity in software development.

If we wish to improve software development, where should we start?

Obviously, it is important to pay attention to software engineering practice, "what" software engineers do (e.g., analyze requirements, create architectural and detailed designs, implement, verify).

However, as in lean manufacturing, it is also critical to focus on software engineering process, "how" software engineers do their work (e.g., plan, organize, measure, make and meet commitments, ensure quality, control, improve, adapt).

Historically, organizations have often tried to implement software process improvement from the top down.

When predictability and quality are perceived (often correctly) to be out of control, a natural management response is to impose standard procedures, approval steps, reporting requirements, and close oversight.

One drawback to this approach is that software development is ultimately knowledge work, very dependent on individual software engineers and teams, who can be the last to be helped by top-down improvement efforts.

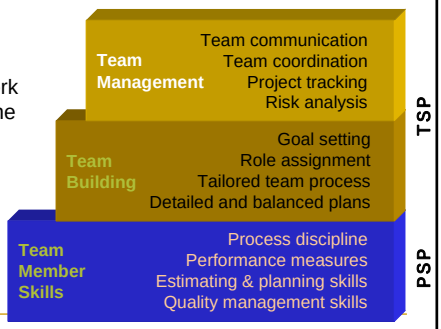
At the Software Engineering Institute (SEI), Watts Humphrey and his colleagues faced this same dilemma.

Humphrey was a major contributor to the development of the software Capability Maturity Model (CMM), which later evolved into the CMMI.

Concerned about the pace of CMM(I) adoption and organizational change, Humphrey asked how these principles could be applied to the work of individual software engineers and teams. This work resulted in the PSP and TSP processes.

The PSP and TSP build capability from the individual software engineer to the development team.

- Improve from the bottom up
  - Where the work is actually done
  - Start with personal process
- Add elements to build and manage the team



Adapted from Davis, 2003

A key feature of SEPI or TSP teams is that they are self-directed.

Teams are responsible for

- **Setting aggressive, achievable goals**
- **Formulating a design and a development strategy**
- **Defining a development process**
- **Making detailed plans and commitments based on historical data**
- **Tracking their own progress and balancing workloads**
- **Identifying, managing, and mitigating risks**
- **Adapting to changes, adjusting plans, developing and selecting alternatives**

Managing self-directed teams requires

- **Clear management and business goals**
- **Mutual trust between management and team**
- **Open communication**

For planning to be meaningful, it must be done in the context of a defined process.

Plan
Design
Design Review
Code
Code Review
Test
Postmortem

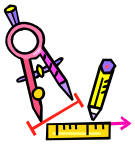
The defined process consists of a sequence of steps; they are phases, not activities.

Scripts specify the process phases, including entry and exit criteria.

Processes based on PSP/TSP principles are biased toward cyclic, incremental delivery.

Historical data implies measurement; in the PSP/TSP, there are four basic process measures.

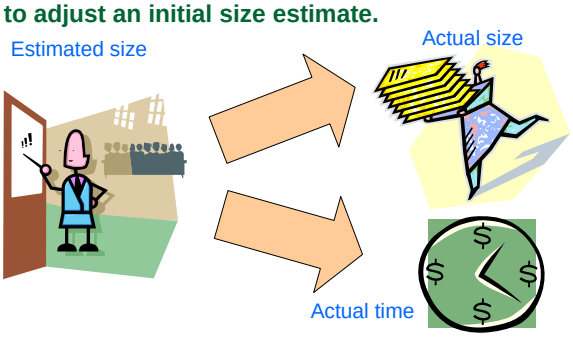
- Size
  - LOC or ??
- Time
  - Minutes by phase
- Quality
  - Defects by phase
- Schedule
  - Task completion date



To improve performance, we must be able to measure and make comparisons.

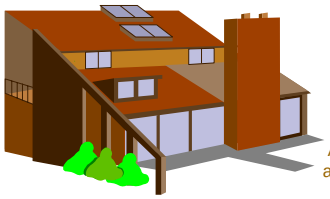
Other measures, such as productivity and defect density, are derived from the base measures.

The planning process projects product size and development time by using historical data to adjust an initial size estimate.



But, how do we make the initial size estimate?

In other domains, it is common to combine the use of direct and proxy measures.

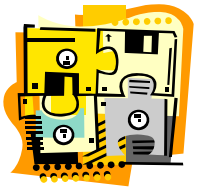


The floor area (square feet) of a house or building often correlates well with the actual construction cost, but may be difficult to visualize in advance.

A proxy measure, like the number and relative size of the rooms, may work better for initial planning.

For software development products, typical proxy measures are classes, web pages, database tables, or document sections.

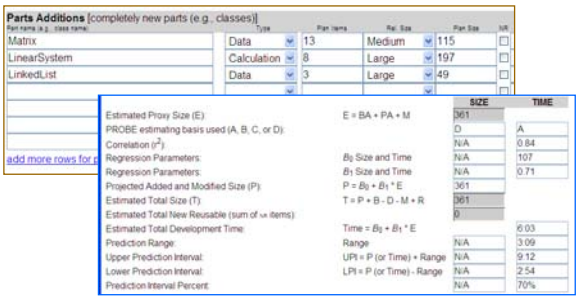
Planning is based on a conceptual design, identifying parts that together could provide the necessary functionality.



Conceptual designs are for planning, and may not represent the final detailed design chosen for a project.

A conceptual design should provide enough guidance for estimating, but not get too deeply into the actual design work.

This planning process often makes use of support tools, such as the open-source Process Dashboard.



Parts Additions (completely new parts (e.g., classes))

Part name (e.g., class name)	Type	Plan items	Rel. Size	Plan Size	IUR
Matrix	Data	13	Medium	115	
LinearSystem	Calculation	8	Large	197	
LinkedList	Data	3	Large	49	

add more rows for p

		SIZE	TIME
Estimated Proxy Size (E):	E = BA + PA + M	361	
PROBE estimating basis used (A, B, C, or D):		D	A
Correlation (r^2):		N/A	0.84
Regression Parameters:	B_0 Size and Time	N/A	107
Regression Parameters:	B_1 Size and Time	N/A	0.71
Projected Added and Modified Size (P):	$P = B_0 + B_1 * E$	361	
Estimated Total Size (T):	T = P + B - D - M + R	361	
Estimated Total New Reusable (sum of ↳ items):		0	
Estimated Total Development Time:	$Time = B_0 + B_1 * E$		6.03
Prediction Range:	Range	N/A	3.09
Upper Prediction Interval:	UPI = P (or Time) + Range	N/A	9.12
Lower Prediction Interval:	LPI = P (or Time) - Range	N/A	2.54
Prediction Interval Percent:		N/A	70%

This proxy estimation projects product size and task time.

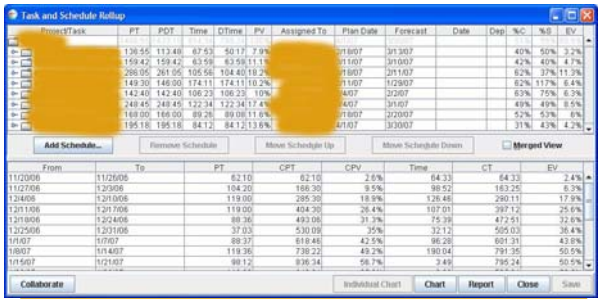
To make a schedule plan, we must know the distribution of available "task time".

Task hours represent the time spent on specific tasks in the project plan.

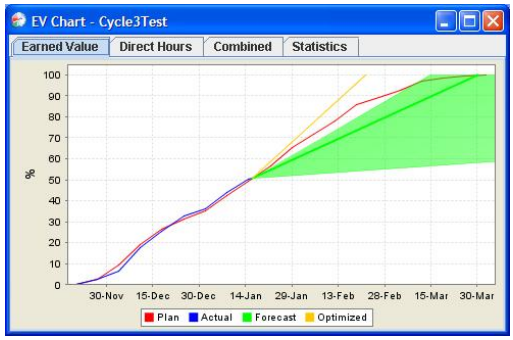
Total work time		
Project		Non-project
Task	Non-task	

How many task hours should a team expect to have in a 40-45 hour week?

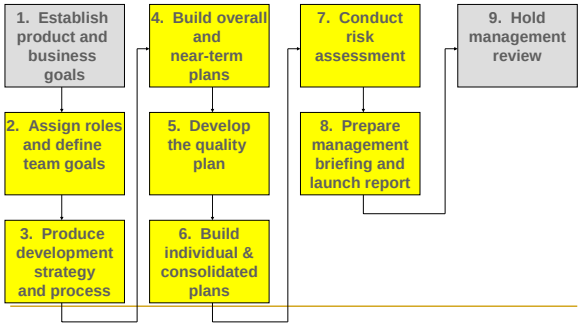
Based on the task and schedule data, task completion dates can be planned and tracked.



Project tracking is facilitated by earned value charts and forecasts.



Team planning and team building is typically done during a week-long "launch".



Although improved planning and tracking is good, we still have to worry about quality.

In SEPI/PSP/TSP, defects are the basic quality measure; does this seem reasonable?

On the one hand, defects are only important to the customer or user if they cause visible adverse effects.

However, defects have a big impact on the software development process, costing money and taking time that could be better spent in adding value for those customers and users.

With the growth of software systems, we are facing a significant quality challenge.

- What if we want 1-10 defects per MLOC?
  - Level 5 CMMI is about 1000 defects/MLOC
- Probably can't "test in" this quality level
- But, probably can't "inspect it in" either:
  - Assume 1000 LOC ≈ 30 pages of listings
  - Then, 1 MLOC ≈ 30,000 pages
  - Can we inspect 30K pages well enough?
    - To leave only 1-10 defects

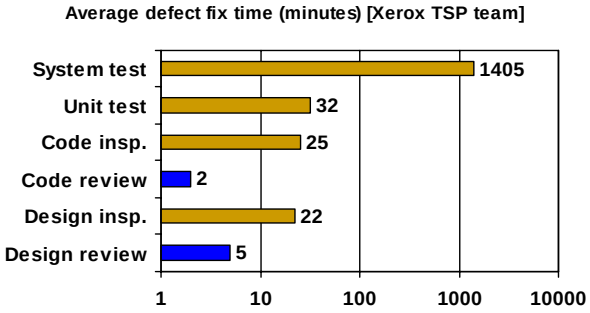
For this reason, quality management must begin with individual software engineers.

The personal process of each software engineer is the place to remove defects, determine their causes, and learn to prevent them.

Engineers and teams use customized design and code review checklists, personal and peer reviews, and team/external inspections.

Using historical data, it is also possible to plan and track quality metrics and to make needed process changes.

Compared to reviews, defect removal in later phases is much more expensive.



Excessive reliance on testing can be a very expensive strategy.

Organizations can use the PSP and TSP to achieve Capability Maturity Model (CMM) goals.

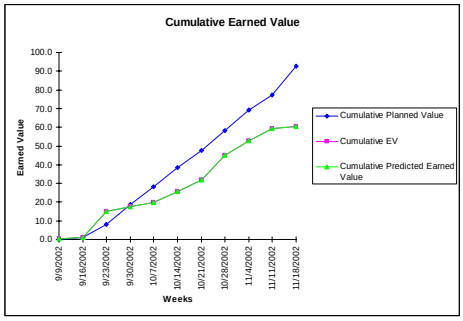
Level	Focus	Key Process Areas (KPA)
5 Optimizing	Continuous process improvement	✓Defect prevention ✓Technology change management ✓Process change management
4 Managed	Product and process quality	✓Quantitative process management ✓Software quality management
3 Defined	Engineering process	✓Organization process focus ✓Organization process definition Training program ✓Integrated software management ✓Software product engineering ✓Intergroup coordination ✓Peer reviews
2 Repeatable	Project management	✓Requirements management ✓Software project planning ✓Software project tracking ✓Software quality assurance ✓Software configuration management Software subcontract management

A similar mapping has been done for CMMI

✓CMM Key Process Area addressed at the project level when using PSP and TSP

Adapted from Willet (SEI), Boston SPIN talk, 2004

Can self-directed teams really solve problem when they arise? A real-world example may help.



Adapted from Willet (SEI), Boston SPIN talk, 2004

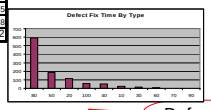
The team analyzed the situation.

Work hours on target

	Plan	Actual	Plan/Actual
Schedule hours this cycle to date	265.2	273.4	0.97
Earned value this cycle to date	69.4	52.6	1.32
To-date hours for tasks completed	199.7	259.9	0.77

Completed tasks show severe underestimates

		Estimate	Actual
module 1	UT	14	26.0
module 2	UT	21	15.8
module 3	UT	4.0	10.5
module 4	UT	4.2	41.8
module 5	UT	6.0	23.5
module 6	UT	8.0	33.8
TOTAL		25.9	151.2



Detail tasks log shows most of problem is in Unit Test

Defect Fix Time by type shows main problem is legacy system defects

Based on the data analysis, the team took corrective action.

- Performed inspections of legacy code
- Looked at ways to increase task hours
- Worked with management to increase the project team size
- Result
  - Team delivered on time
  - A defect-free product
  - Increased cost – management made decision

Since spring 2005, MSOE has been working with industry partners on the SEPI effort.

- SEPI engineer course – 10 days
  - Three offerings, 56 engineers, 3 organizations
- SEPI management course – 2 days
  - Three offerings, ~30 managers, 3 organizations
- SEPI projects
  - Seven projects, 9 teams, ~37 engineers
    - Positive engineer/team and management feedback

Here are a few references on software engineering process, PSP, and TSP.

- W. Humphrey, *PSP – A Self-Improvement Process for Software Engineers*, A-W, 2005
- W. Humphrey, *TSP – Leading a Development Team*, A-W, 2006
- W. Humphrey, *TSP – Coaching Development Teams*, A-W, 2006
- W. Humphrey, *Introduction to the Team Software Process*, A-W, 2000
- W. Humphrey, *Winning with Software: An Executive Strategy*, A-W, 2002
- N. Davis & J. Mullaney, *The Team Software Process in Practice: A Summary of Recent Results*, SEI, 2003, CMU/SEI-2003-TR-014 (www.sei.cmu.edu/tsp/)
- J. McHale & D. Wall, *Mapping TSP to CMMI*, SEI, 2005, CMU/SEI-2004-TR-014 (www.sei.cmu.edu/tsp/)
- M. Sebern, "Software process: applying industrial-strength methods in engineering education", *Proceedings of ASEE Annual Conference*, 2005, American Society for Engineering Education.

Questions?