The Personal Software Process Strategy

An Overview

Outline

- PSP Overview
- SW Process & Process Maturity
- The 5-Level CMM
- PSP Levels
- Logic & Principles of the PSP
- Productivity

PSP Overview

PSP Definition

- "The personal software process (PSP) is a self-improvement process designed to help you control, manage, and improve the way you work. It is a structured framework of forms, guidelines, and procedures for developing software. Properly used, the PSP provides the historical data you need to better make and meet commitments and it makes routine elements of your job more predictable and more efficient."
  (Humphrey, 1995, p. 1)

PSP Strategy (cf. Humphrey, 1995, p. 9)

- Identify large-system SW methods / practices which can be used by individuals
- Define subset of these that can be applied while developing small programs
- Structure these so they can be gradually learned
- Provide exercises for learning these methods / practices

The PSP is not magic!

(cf. Humphrey, 1995, p. 2)

- The PSP is not magic.
- It can be a great help in guiding your SW development.
- But you must make the improvements it suggests yourself.

Why do we need a SE discipline? (Humphrey, 1995, p. 2, 3)

- Most software development:
  - Does not follow a defined process
  - Is performed intuitively
  - Requires extensive testing and repair
  - Uses unpredictable, non-repeatable processes
- This is similar to Brownian Motion
  - Individual particles of gas cannot be predicted.
  - The entire volume can be described statistically.

Need for SE Discipline (cont.)

- Following this analogy:
  - Large-scale SW development is treated as "crowd control":
  - One doesn't worry about what each individual does,
  - Just what the overall process is like.
- This approach is:
  - Expensive
  - Error-prone
  - Risky
- We need:
  - More detailed methodologies
  - Standards
  - Frameworks

Need for SE Discipline (cont.)

- Examples of disasters
  - Thorac
    - Administered too much radiation (no safety control) and killed two patients.
  - Telephone system
    - Switching system failure shut down large segments of the U.S.
  - Power system
    - Overload carried over and affected multiple adjacent grids and "blacked out" much of Northeast.

A Disciplined SE Organization

(Humphrey, 1995, p. 3)

- In order to address these issues, a disciplined SE organization will:
  - Have well-defined practices.
  - Strive to continually improve these practices.

SW Process & Process Maturity

What is Software Process?

(Humphrey, 1995, p. 4, 5)

- A "software process is the sequence of steps required to develop or maintain software… [It] sets out the technical and management framework for applying methods, tools, and people to the software task."

Software Process Definition

(Humphrey, 1995, p. 5)

- A "software process definition is a description of [the SW development] process. When properly designed and presented, the definition guides software engineers as they work…
- [It] identifies roles and specifies tasks… establishes measures and provides exit and entry criteria for every major step.
- An effectively designed definition helps to ensure that every work item is properly assigned and its status is tracked.
- It also provides an orderly mechanism for learning. As better methods are found, they are incorporated into the organization's official process definitions. A defined process thus permits each new project to build on its own experiences as well as its predecessors'."
- cf. Kellner's list of benefits of operationally defining a process. (Humphrey, p. 5)

Process Maturity (Humphrey, 1995, p. 6)

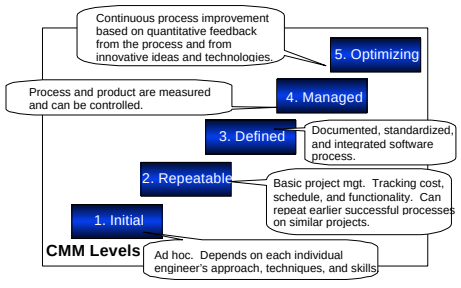
- An organization's level of software development process maturity indicates the following about its development process:
  - How well defined it is
  - How repeatable
  - How well managed
  - Whether it is optimizing / improving

The 5-Level CMM

CMM Overview

- CMM = Capability Maturity Model
  - An orderly way for organizations to determine the capabilities of their current processes and to establish priorities for improvement.
  - 5 Levels

5 Levels of the CMM (cf. Humphrey, 1995, p. 6, 7)



Key Process Areas of the CMM

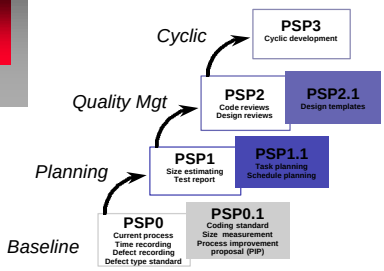
(cf. Humphrey, 1995, p. 7)

CMM Level	Key Process Areas
1. Initial	• Ad Hoc, Chaotic • Process dependent on each individual developer
2. Repeatable	• SW Config Mgt • SW QA • SW Subcontract Mgt • SW Proj Tracking & Oversight • SW Proj Planning • Req's Mgt
3. Defined	• Peer Reviews • Intergroup coordination • SW Prod Engineering • Integrated SW Mgt • Training • SW Process Definition • SW Process Focus
4. Managed	• Quality Mgt • Quantitative Process Mgt
5. Optimizing	• Process Change Mgt • Technology Change Mgt • Defect Prevention

PSP Levels

Overview of PSP Levels

(Humphrey, 1995, p. 11)



The Baseline Personal Process

(cf. Humphrey, 1995, p. 9-11)

- *PSP 0*
  - *Current process*
  - *Time recording*
  - *Defect recording*
  - *Defect type std*
- *PSP 0.1*
  - *Coding std*
  - *Size measurement*
  - *Process improvement proposal*

The Personal Planning Process

(cf. Humphrey, 1995, p. 11)

- *PSP 1*
  - *Size estimating*
  - *Test report*
- *PSP 1.1*
  - *Task planning*
  - *Schedule planning*

Personal Quality Mgt

(cf. Humphrey, 1995, p. 11, 12)

- *PSP 2*
  - *Code reviews*
    - *Uninspected code: 50-200 defects / KLOC*
    - *Inspected code: 75-200 defects / KLOC*
    - *The # found in compile & test decreases, but the total increases. IT IS MUCH LESS COSTLY TO FIND DEFECTS EARLY, so the increase in defect rate is GOOD!*
  - *Design reviews*
- *PSP 2.1*
  - *Design templates*
    - *specification of what must be in a completed design*

Cyclic Personal Process

(cf. Humphrey, 1995, p. 11, 13)

- *PSP 3*
  - *Cyclic development*
    - *Quality assumed in earlier iterations*
    - *Regression testing*
    - *Can develop programs with several 10's KLOC. (This is too large for PSP 2 to handle.)*

The Team Process *(cf. Humphrey, 1995, p. 13, 14)*

- *TSP*
  - *help, support, review*
  - *PSP can help make make more effective team members*

Logic and Principles of the PSP

Logic of the PSP *(cf. Humphrey, 1995, p. 14)*

1. *SW professionals will better understand what they do if they define, measure, and track their work.*
2. *They will then have a defined process structure and measurable criteria for evaluating and learning from their own and others' experiences.*
3. *With this knowledge & experience, they can select those methods and practices which best suit their particular tasks and abilities.*
4. *By using a customized set of orderly, consistently practiced, and high-quality personal practices, they will be more effective members of their development teams and projects.*

PSP's Underlying Principles

(cf. Humphrey, 1995, p. 14-19)

1. *Defined / structured process can improve working efficiency.*
   - *There are both creative & repetitive parts of work.*
   - *Just because some parts of work are creative is no reason to treat them all that way.*
2. *Personal processes need to fit individual preferences.*
3. *SW professionals should define their own processes and continually refine them.*
4. *Processes should evolve with users to meet changes in industry, etc.*
5. *Feedback enhances process improvement.*

Productivity

Changes in Productivity Associated with Using the PSP *(cf. Humphrey, 1995, p. 19)*

- *Productivity may initially DECREASE while one is in the process of learning the PSP.*
- *Eventually productivity should INCREASE after the PSP techniques become habits.*
- *Hopefully productivity will increase beyond your initial level over time.*

Variation in Personal Productivity *(cf. Humphrey, 1995, p. 19-21)*

- *Amount of time to develop a program is only one measure of the quality of the work done.*
- *Even given a set programs which are all of high quality and are developed by different engineers there will still be (possibly wide) variation in the amount of time taken and the size of each program.*
- *Some influencing factors:*
  - *User messages included*
  - *Error checking performed*
  - *Program generalization*
  - *…*
- *cf. figs. 1.6 - 1.9, p. 20-22*

Caveats about the PSP

Caveats *(cf. Humphrey, 1995, p. 25)*

- *This course / book concentrates on design, code, and test. The PSP may be applied to other aspects of SW development too (req's spec, maint, test planning, etc.)*
- *Tools for supporting the PSP are not discussed. Obviously they could be beneficial. But it is best to first learn the process by hand so that you understand and can better customize the process.*
- *A combination of defined process, tools, and learning / improvement perspective is better than any one approach alone (i.e. a "systems" perspective).*