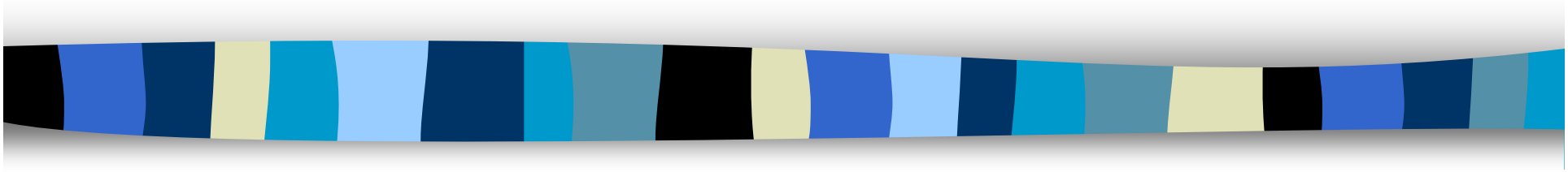
INF 111 / CSE 121: Software Tools and Methods



Lecture Notes for Fall Quarter, 2007

Michele Rousseau

Set 25

Announcements

- **Quiz #4 - Is available**
- **Final Review on Friday**

Previously in INF 111…

- **Effort Estimations**
  - Algorithmic Cost Models
  - COCOMO

Today's Lecture

- **Effort Estimation**
  - Algorithmic Cost Modeling
    - COCOMO
- **Personal Software Process (PSP)**

COCOMO: Some Assumptions

COCOMO => “COnstructive COst Model”

- Primary cost driver ➔DSI
  - Delivered Source Instructions (DSI) developed by the project
  - Only code developed by staff
  - Excludes
    - Test drivers & other support code
    - Comments
    - Declarations
    - Code developed by application generators
  - SLOC => Single logical line of code → eg. If;then;else

COCOMO: Three Models

- **3 Models reflect the** complexity**:**
  - the Basic Model
  - the Intermediate Model
  - and the Detailed Model

The Development Modes: Project Characteristics

- Organic Mode
  - developed in a **familiar**, stable environment,
  - **similar** to the previously developed projects
  - relatively **small** and requires little innovation
  - Eg. Payroll system
- Semidetached Mode
  - **intermediate** between Organic and Embedded
  - Eg. Banking System
- Embedded Mode
  - tight, **inflexible** constraints and interface requirements
  - The product requires **great innovation**
  - Eg. Nuclear power plant system

Intermediate COCOMO Model

Estimates effort by using fifteen cost driver variables besides the size variable used in Basic COCOMO

- When should you use it?
  - Can be applied across the entire software product for easy and rough cost estimation during the early stage
  - or it can be applied at the software product component level for more accurate cost estimation in more detailed stages

Cost Drivers

Four areas for drivers

- **Product Attributes**
  - Reliability, Database Size, Complexity
- **Computer Attributes**
  - Execution Time Constraint, Main Storage Constraint, Virtual Machine Volatility, Computer Turnaround Time
- **Personnel Attributes**
  - Analyst Capability, Applications Experience, Programmer Capability, Virtual Machine Experience, Programming Language Experience
- **Project Attributes**
  - Modern Programming Practices, Use of Software Tools, Required Development Schedule

Subjective Assessments

Intermediate Model: Effort Multipliers

- Table of Effort Multipliers for each of the Cost Drivers is provided with ranges depending on the ratings

Cost Driver	*Very Low*	*Low*	*Nom*	*High*	*Very High*	*Extra High*
Product Complexity	**0.70**	**0.85**	**1.00**	**1.15**	**1.30**	**1.65**

Intermediate Model: Equations

Mode	*Effort*	*Schedule*
Organic	$E=EAF*3.2*(KDSI)^{1.05}$	$TDEV=2.5*(E)^{0.38}$
Semi-detached	$E=EAF*3.0*(KDSI)^{1.12}$	$TDEV=2.5*(E)^{0.35}$
Embedded	$E=EAF*2.8*(KDSI)^{1.20}$	$TDEV=2.5*(E)^{0.32}$

COCOMO Effort Equation

Effort = 3.0 * EAF * $(KSLOC)^E$

- Result is in Man-months
- EAF → Effort Adjustment Factor
  - Derived from Cost Drivers
- E → Exponent
  - Derived from five scale drivers
    - Precedentedness
    - Development Flexibility
    - Architecture / Risk Resolution
    - Team Cohesion
    - Process Maturity

Intermediate Model: Example

- Project A is to be a **32,000 DSI semi-detached software**. It is in a mission critical area, so the **reliability** is high (RELY=high=1.15).

Then we can estimate:

- **Effort** = $1.15*3.0*(32)^{1.12}$ = 167 man-months
- **Schedule** = $2.5*(167)^{0.35}$ = 15 months
- **Productivity** = (DSI / MM) = 32,000 DSI/167 MM
  = 192 DSI/MM
- **Average Staffing** = = 167 MM/15 months
  MM/Schedule Months = 11 FSP

Intermediate Model: Limitations

- Estimates are within **20%** of the actuals **68%** of the time
- Its effort multipliers are phase-insensitive
- It can be very tedious to use on a product with many components

Detailed COCOMO Model: How is it Different?

- **Phase-sensitive Effort Multipliers**
  Effort multipliers for the cost drivers are different depending on the software development phases
- **Module-Subsystem-System Hierarchy**
  - The software product is estimated in the three level hierarchical decomposition.
  - The fifteen cost drivers are related to module or subsystem level

Detailed COCOMO Model: When Should You Use It?

- The Detailed Model can estimate
  - **the staffing, cost, and duration of each of the development phases, subsystems, modules**
- It allows you to experiment with different development strategies, to find the plan that best suits your needs and resources

Detailed Model: Equations

- Same equations for estimations as the Intermediate Model
- Uses a very complex procedure to calculate estimation.
  - **The procedure uses the DSIs for subsystems and modules, and module level and subsystem level effort multipliers as inputs**

Detailed Model: Limitations

- Requires substantially more time and effort to calculate estimates than previous models
- Estimates are within 20% of the actuals 70% of the time

COCOMO II

- **Modified for more current development**
- **3 increasingly detailed cost estimation models**
  - **Application composition**
    - **Prototyping efforts (UI Issues)**
    - **Used in a powerful CASE environment**
  - **Early Design**
    - **Focused on Architectural design phase**
  - **Post-Architecture model**
    - **Used during implementation phaseCOCOMO estimates assume good mgmt**
- by both the developer and the customer

- Assumes the requirements specification is not substantially changed after the requirements & design phase

- **http://sunset.usc.edu/research/COCOMOII/index.html**

Data Collection

- Regardless of the method or model used, data is needed for calibration
- Programmers need to know their own "constant adjustment factors"
  - **Goal of Personal Software Process to establish such a database**

Overview of PSP

The Personal Software Process (PSP)

- **PSP sets out the principal practices for defining, measuring and analysing an individual's own processes**

- **The main idea:**
  - understand how you work
  - analyze your performance
  - Improve your process
  - Develop an ability to define, measure and analyze your process

PSP

- **PSP applies a CMM-like assessment for individual work**
  - Measurement & analysis framework to help you characterize your process
    - Self-assessment and self-monitoring
  - Prescribes a personal process for developing software
    - defined steps
    - Forms
    - Standards
  - Assumes individual scale & complexity
    - Well-defined individual tasks of short duration

PSP - Steps

1. **Understand the current status of your development process or processes.**
2. **Develop a vision of the desired process.**
3. **Establish a list of required process improvement actions, in order of priority.**
4. **Produce a plan to accomplish the required actions.**
5. **Commit the resources to execute the plan.**
6. **Start over at step 1.**

PSP Overview

- **The PSP is introduced in 7 upward compatible steps (4 levels)**
- **Write 1 or 2 *small* programs at each step**
  - Assume that you know the programming language
- **Gather and analyze data on your work**
  - Many standard forms & spreadsheet templates
- **Use these analyses to improve your work**
  - Note patterns in your work

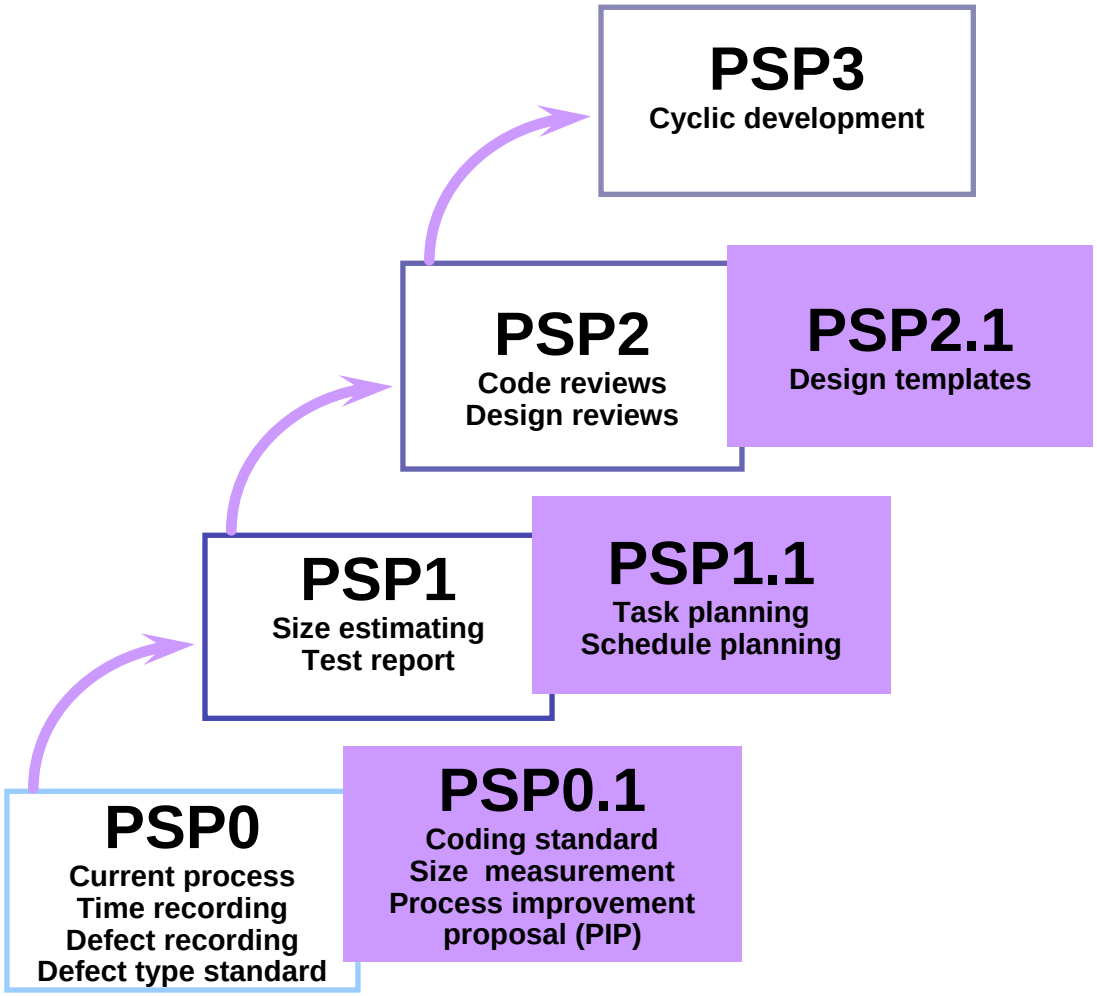
PSP Evolution

Cyclic Personal Process

Personal Quality Management

Personal Planning Process

Baseline Personal Process



Why use PSP?

- **demonstrates personal process principles**
- **assists engineers in making accurate plans**
- **determines the steps engineers can take to improve product quality**
- **establishes benchmarks to measure personal process improvement, and**
- **determines the impact of process changes on an engineer's performance."**

PSP Evaluation

- **Humphrey has used in SE courses**
  - Improvements in time-to-compile, quality and productivity
- **Patchy, but promising use in industry**
  - E.g. Nortel (Atlanta)
- **Still immature**
- **Requires large overhead for data gathering**
  - Not clear that you should use permanently or continually

PSP/TSP/CMM

CMM® Builds organizational capability

TSP℠ Builds quality products on cost and schedule

PSP® Builds individual skill and discipline