

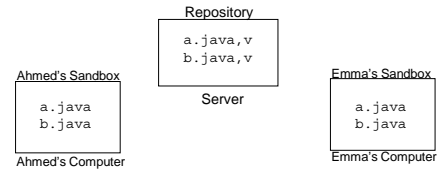
Version Control Tools

- Provide a complete history of a file
 - All current and previous versions available- nothing ever goes away
- Other names
 - Revision Control System
 - Source Control
 - Configuration Management
- Idea:
 - Keep files in repository
 - Also keep a history of the changes to the file
 - Use the history to recover any version of the file
 - Also record times: "What did these files look like at noon last Friday?"
 - Can also tag groups of files

Week 3 Lecture Slide 1

How Version Control Works

- Place the official version of source code into a central repository, or database
- Programmers check out a working copy into their personal sandbox
- When finished and fully tested, programmers check in their code back to the repository



Week 3 Lecture Slide 2

CVS

- Concurrent Version System
 - UNIX tool, Open Source
 - Available since 1989
 - Suitable for individuals or medium-sized teams
 - <http://www.cvshome.org>
- On UNIX, check in and check out accomplished by typing commands
- GUI front ends/clients available
 - WinCVS, Eclipse

Week 3 Lecture Slide 3

CVS Commands

- checkout
 - Retrieving a file from the repository
- commit
 - Putting changes back into repository
- update
 - Refresh the local or sandbox copy with any changes since checkout
- CVS Branches vs. Eclipse Branches

Week 3 Lecture Slide 4

Control Regimes

- Pessimistic Model
 - File becomes locked on check out
 - Nobody else can make changes
- Optimistic Model
 - File not locked on check out
 - Conflicts or collisions are managed at check in
- Default in CVS is optimistic model, but pessimistic model available

Week 3 Lecture Slide 5

Conflict Detection and Management

- On check in, official repository copy is compared with new copy
 - Check version number
 - If repository and sandbox versions are the same, accept the changes
 - If repository version is newer, reject commit operation.
- Need to update sandbox before check in
 - Includes a synchronization step to merge changes from two files (new repository version and modified file from sandbox)

Week 3 Lecture Slide 6

Conflict Detection and Management

- Merge algorithm
 - Line by line comparison
 - Changes to different lines are OK
 - Changes to same lines labeled as a conflict
 - Both versions written to the sandbox copy
 - Choose which line by editing manually
 - No guarantee of code correctness after merge
- On a successful check in, save new version of files with a set of backwards references to changes
- Can apply detection selectively
 - Binary files only use version numbers or timestamps
 - No conflict detection applied to files that are ignored

Week 3 Lecture Slide 7

What to check in

- Don't check in files that are automatically created from others
 - e.g. .class files
 - Uses disk space and bandwidth for no reason
- Do check in:
 - Your own little test programs
 - And their expected output
 - Readme files, notes, build logs, etc.
 - Anything else you created by hand

Week 3 Lecture Slide 8

When to check in

- Version control is not a backup system
 - Your computer should have one of those
- Don't check in just because you're taking a break to surf the web
- Check in files when they are stable
 - e.g. After adding a new feature
- Or when you have to move from location to location
 - e.g. From home to school or vice versa

Week 3 Lecture Slide 9

Tagging

- Use tags to label a group of files
 - Makes it easy to check out a release or configuration



- See Eclipse Help topic "Discovering branch and version tags"
- Don't use stickies (tag, revision, keyword)

Week 3 Lecture Slide 10

Branching

- Using a particular version as the baseline for a series of versions
- Reasons for branching
 - Variations on a theme
 - Experimental code
 - Bug fix chains
- Happens at the repository, not your sandbox

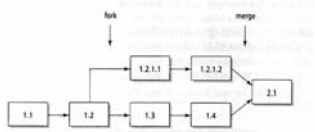


Figure 19.6 Forking and merging of development paths

Week 3 Lecture Slide 11

Final Details on CVS

- Batch mode synchronization vs. interactive synchronization
- Keyword expansion
- Watches
 - Requires modification
 - "Watch" sets how you want to be notified
 - "Edit" asks to be notified
 - See Eclipse Help topic "Finding out who's working on what: watch/edit"

Week 3 Lecture Slide 12

Keyword Expansion

- CVS replaces strings like `$(Keyword):$` with associated value during check-in


```

$Author: frost $
$Date: 2005/01/11 14:54:32 $
  - Always in Universal Time
$Revision: 1.1 $
            
```
- Often embed these values in strings, rather than in comments


```

public static final String VERSION =
"$Revision: 1.1 $";
public String getVersion() {
    return VERSION.split(' ')[1].trim();
}
            
```

Week 3 Lecture Slide 13

Personal Software Process

- Something in between CMM and XP
- Developed by Watts S. Humphrey starting in mid-1980s
 - Watt S. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, 1995.
- Describes a defined, measured, planned and quality-controlled process for individuals
- Influenced by
 - Humphrey's background as a physicist
 - Frederick Winslow Taylor's principles of task analysis, scientific management, or industrial engineering from 1880s

Week 3 Lecture Slide 14

Effort Estimation and Planning

- Predicting the resources required for a software development process
- How much effort is required to complete an activity?
- How much calendar time is needed to complete an activity?
- What is the total cost of an activity?
- Project estimation and scheduling and interleaved management activities

© 2000 Ian Sommerville

Week 3 Lecture Slide 15

Missing the Mark

- As a group, the software industry is notoriously bad at effort estimation
 - Over budget, over schedule
 - Incomplete, buggy products
 - Release dates pushed back by months
- Brooks, Chapters 8 and 14 (and others)
- Reasons
 - Uncertainty and unknowns about project details
 - Don't have or use good techniques
 - Good techniques are applied poorly
 - Practice and historical databases
- Origins of large-scale failures can be found in your own work habits

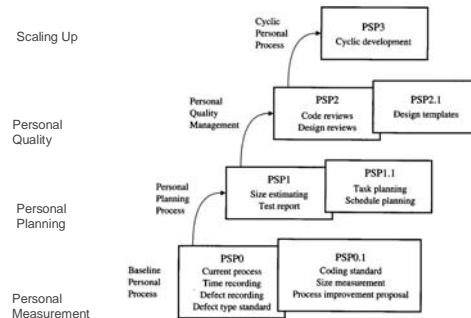
Week 3 Lecture Slide 16

PSP Training

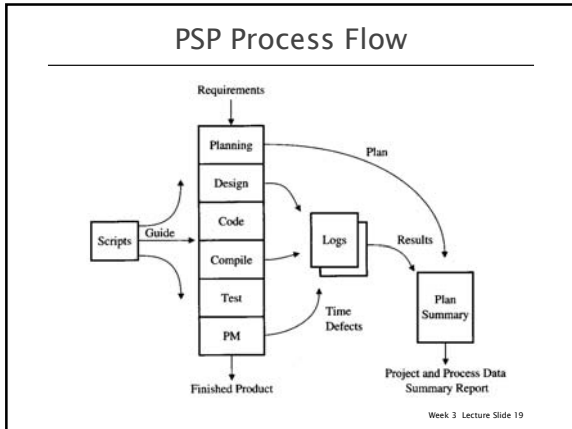
- A series of 10 exercises and 5 reports are used to illustrate PSP principles
 - Exercises are small programs, most about 100 LOC, last two 200-300 LOC
- Objective is to teach developers to:
 - Measure their work
 - Analyze these measures
 - Set and meet improvement goals
 - Plan work
- Dovetails with CMM

Week 3 Lecture Slide 17

Seven PSP Principles



Week 3 Lecture Slide 18



Time Recording Log

TABLE 2.7 TIME RECORDING LOG EXAMPLE

Student Student 3 Date 1/19/94
 Instructor Humphrey Program # 1A

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
1/19	16:25	16:30	0	5	plann'g	
	16:35	17:05	0	30	design	
	17:05	17:40	3	32	code	phone call
	17:40	17:55	0	15	compile	
	17:55	18:00	0	5	test	
1/21	9:25	9:30	0	5	pm	
1/24	15:40	15:55	0	15	pm	
	17:15	17:25	0	10	pm	

Week 3 Lecture Slide 20

Defect Recording Log

2.7 Defect Recording Log 47

Defect Types

10 Documentation	60 Checking
20 Startup	70 Data
30 Build Package	80 Function
40 Assignment	90 System
50 Interface	100 Environment

TABLE 2.10 DEFECT RECORDING LOG EXAMPLE

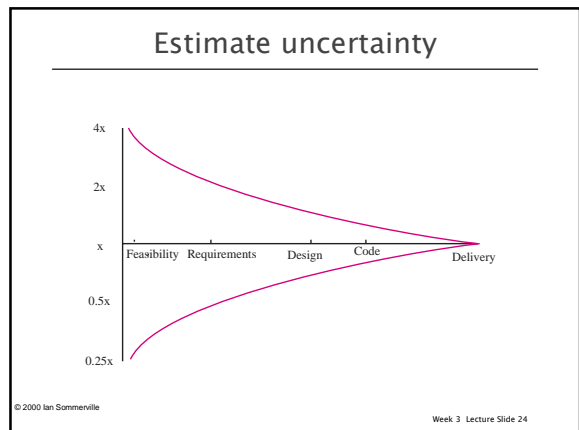
Student Student 3 Date 1/19/94
 Instructor Humphrey Program # 1A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
		EC	code	compile	1:00	
Description: missing semicolon						
		EC	code	compile	1:00	
Description: missing semicolon						
		EC	code	compile	1:00	
Description: wrong type on RHS of binary operator, must cast integers as float						
		EC	code	compile	1:00	
Description: wrong type on RHS, constant literal should be 0.0 not 0						

Week 3 Lecture Slide 21

- ### Results from PSP
- Study of 104 students
 - 58% decrease in average number of defects injected
 - 71.9% decrease in number of defects found during testing
 - 20.8% increase in LOC/hr
 - Estimation and planning accuracy increased
- Watts S. Humphrey, "Using A Defined and Measured Personal Software Process," IEEE Software, May, 1996, pp. 77-88
 Week 3 Lecture Slide 22

- ### Estimation on Assignments
- Two techniques used come from my research
 - Conducted field studies of software companies and these are the ones that worked best
 - Pulls together ideas from many sources
 - Assignment 1: Naïve estimation
 - Take your best guess
 - Assignment 2: Estimation by parts
 - Bottom-up or top-down, depending on where you start
 - Assignment 3: Re-estimation
 - As more time is spent on a project, uncertainty decreases
 - Practice and better techniques result in better estimates
- Week 3 Lecture Slide 23



Productivity Rates

- Longhorn Project (2003)
 - 16 MLOC, 5000 people, 3 years
 - 1067 LOC/person/year
- Grady and Caswell at HP (1987)
 - ~1100 LOC/person/year
- Brooks (1975)
 - IBM OS/360
 - 600-800 instructions/person/year in control group
 - 2000-3000 instructions/person/year in language translator group
 - Corbató's Data
 - 1200 lines/person/year (PL/I)

Week 3 Lecture Slide 25

Factors Affecting Productivity Rates

- Application domain experience
- Process quality
- Project size
 - Negative relationship
- Technology support
- Working environment

Week 3 Lecture Slide 26

Chamber Groups vs. Orchestra

- Chamber Groups
 - Everyone watching and listening to each other constantly
 - Takes turns with the lead
 - One part per instrument
- Orchestra
 - Everyone watching the conductor
 - Complaining about everybody else
 - Conductor is leading, but different sections are "prime"
 - Multiple instruments per part

Week 3 Lecture Slide 27

Team Organization

- Homogeneous Peers
 - Everyone does everything
- Hierarchical Organization
 - Chief Programmer Team
 - See also The Surgical Team, Brooks Chapter 3
- Matrix Organization
- SWAT Team
- Open Structured Team

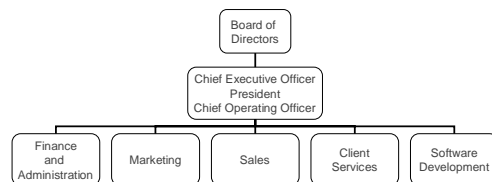
Week 3 Lecture Slide 28

Homogeneous Peers

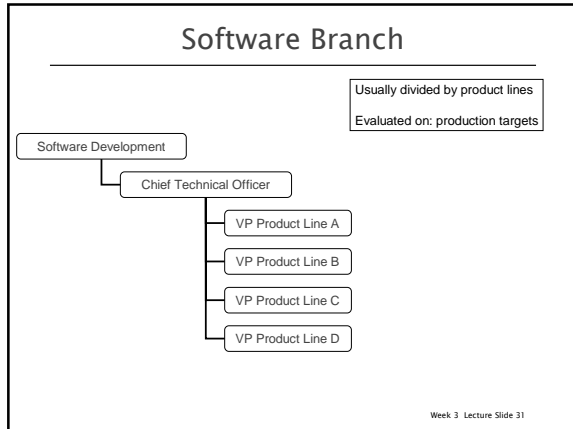


Week 3 Lecture Slide 29

Hierarchical Organization



Week 3 Lecture Slide 30



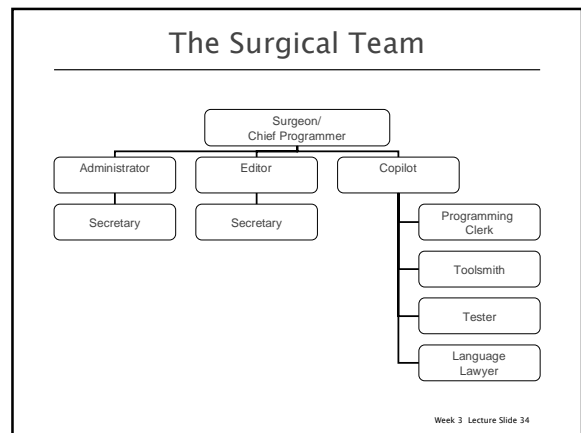
Matrix Organization

- Basic unit is small specialized group
- Projects are formed with people from units with different specialties

	Real-time	Graphics	DB	QA	testing
Project A	X			X	X
Project B	X		X	X	X
Project C		X	X	X	X

Week 3 Lecture Slide 32

- ### The Surgical Team
- Ideal: Small, sharp teams of the best (<10)
 - Too slow for large systems
 - Examples: Any software you've ever bought
 - Example: OS/360, 5000 person-months over 3 years
 - Increased team size, increased communication costs, decreased conceptual integrity
 - Proposal: The surgical team organized around a Chief Programmer
 - Specialized tasks, supporting personnel
 - Goal is to leverage a scarce resource: a highly talented and experienced developer
- Week 3 Lecture Slide 33



- ### Members of The Surgical Team
- Surgeon/Chief Programmer
 - Senior developer with 10+ years of experience and talent
 - Solid expertise in programming, systems, and problem domain
 - Personally designs, codes, tests, and documents the program
 - Copilot
 - Acts as backup to the surgeon (alter ego)
 - Like surgeon, but with less experience
 - Thinks about, discusses, and evaluates surgeon's design
 - Researches alternatives
 - Intimately familiar with source code
- Week 3 Lecture Slide 35

- ### Members of The Surgical Team
- Program Clerk
 - Maintains all technical records
 - Trained as a secretary
 - Manages paperwork/files: inputs to computer, program listings, data libraries
 - Controls integrity and availability of program
 - Toolsmith
 - Installs or creates tools needed by surgeon
 - Libraries of utilities, software tools (e.g. source code searching)
 - Complements system administrators
- Week 3 Lecture Slide 36

Members of the Surgical Team

- Tester
 - Creates test plans
 - Builds testing harnesses, writes test cases, runs tests
- Language Lawyer
 - Knowledgeable about syntax and standards for a programming language
 - Make recommendations on technique
 - "...can find a neat or efficient way to use the language to do difficult, obscure, or tricky things."
 - Caveat: Doesn't pay to be too clever
 - Can serve more than one team at once

Week 3 Lecture Slide 37

Members of the Surgical Team

- Administrator
 - Looks after money, space, machines, people, so Surgeon can focus on programming
 - Depending on complexity of project, can serve more than one or two teams
 - Supported by a secretary
- Editor
 - Writes and edits documentation
 - Works from drafts or notes from surgeon
 - Responsible for final production
 - Supported by a secretary

Week 3 Lecture Slide 38

Comments on the Surgical Team

- Effect of email on division of labor
 - Communication is more dynamic and informal
 - Harder to insulate chief programmer
- No succession plan
 - Where do chief programmers come from?
 - Where's the farm team?
- Roles on the Surgical Team has strong resemblances to roles in modern software development

Week 3 Lecture Slide 39

Software Development Roles

- Developers
 - Design, code, test. Involved in many aspects.
- Project management
 - Manage development schedule, interfaces to rest of organization.
 - Change control board: Manages risk of changes made to shipping product
 - SEPG/SPI: Organization-wide advisers on software process improvement
 - Reuse board: Manages organization-wide library of reusable components
- UI designers
 - Study usage, specifies UI, designs and mocks-up screens and interactions

Week 3 Lecture Slide 40

Software Development Roles

- Testers
 - Test the product, supports development
- Data wranglers and build masters
 - Manage versions of code, supervises builds
- Release planning
 - Manage collaborative development tools, executes process steps to package and ship product
- Topic experts (e.g., security, databases)
 - Design and/or implement specific aspects of product
- Technical writers
 - Write user documents, helps with technical documents
- Graphic designers: images, branding, packaging

Week 3 Lecture Slide 41

General Principles

- User fewer, and better, people.
- Try to fit tasks to the capabilities and motivation of the people available.
- In the long run, an organization is better off if it helps people to get the most out of themselves.
- It is wise to select people such that a well-balanced and harmonious team results.
- Someone who does not fit the team should be removed.

Week 3 Lecture Slide 42

