

CS 350, slide set 2

M. Overstreet
Old Dominion University
Spring 2005

Reading

- PSP text, ch. 4, 5, 6

Planning

- Period plans
 - How shall I spend my time?
 - Schedule oriented
- Product plans
 - What are we going to do over the next several weeks?
 - Activity oriented
- For any project, these 2 plans are coupled

Weekly Activity Summary

- First step in period planning:
 - Understand how you spend time currently
- Start with Time Recording Log
- Transfer to Weekly Activity Summary

Example: time log

date	start	stop	int. time	delta	activity	comments	C	U
1/26	2:00 p	4:25 p	15.5	125	Read T	Read chap 4, 5	x	2
1/27	5:45 p	7:05 p		80	Class	Lect, chap. 4, 5	x	1
1/28	8:45 p	10:20 p	3,3,4	95	Read S	Read Prog 1 req.		
1/28	10:00 p	10:30 p		30	Prog	Prog 1 design		
1/28	10:30 p	11:05 p		35	Read T	Read chap. 6	x	1
1/30	5:30 p	6:35 p	5	55	Prog	Code prog. 1		
1/30	8:45 p	9:20 p	3	30	Prog	Test data for p.1		
1/30	9:30 p	9:50 p		20	Prog	Code prog. 1		
1/30	9:50 p	10:25 p		35	Prog	Comp & test p. 1	x	1
1/29	5:45 p	7:00 p		75	Class	Lect. ch 5 & 6	x	1
1/31	4:00 p	4:45 p		45	Read T	Read chap. 7	x	1

Example Weekly Activity Summary

Task Date	Class	Write Prog	Quiz Prep	Read Text	Study Spec	Total
Sun 1/26				125		125
Mon	80					80
Tu		30		35	95	160
Wed	75					75
Th		140				140
Fri						
Sat				45		45
Totals	155	170		205	95	635

Comments: 1. numbers made up
 2. recall rule-of-thumb: spend 2 hrs out of class for each hr in class

Example (cont.)

Task	Class	Wrt P	Quiz	Read T	Read S	Total
Previous Week's Time						
Total	150	0	25	145	0	420
Avg.	150	0	25	145	0	420
Max.	150	0	25	145	0	420
Min.	150	0	25	145	0	420
Current Week's Times						
Total	305	170	25	360	95	1055
Avg.	152	170	25	175	95	522
Max.	155	170	25	205	95	635
Min.	150	170	25	145	95	420

Assume we have only one week.

Semester total, avg, max & min!

Ignore 0's!

Assignment:

- Pg. 43.
- Due next Wed., Jan. 26.
 - Must mail to cmo@cs.odu.edu

Product Planning

- Some definitions
 - **Product:** something you produce, usually for someone else
 - **Project:** produces a product
 - **Task:** an element of work
 - **Process:** the steps to produce a product
 - **Plans:** the way a specific project is to be done; how, when, who, at what costs
 - **Job:** something you do; either a project or a task

Job number log

- Used to record estimated and actual times
- Used for product planning
- Goal: to have personal data on how long certain types of activities take
 - Must be based on size
 - Maybe lots of other factors, but we start with size
 - Used as key input for future time estimates

Job Number Example

- See detailed example, pg. 50 based on time log, pg. 51
- Detailed instructions, pg. 52
- See detailed term def'ns, pg. 53

Partial example: programming data only

Job #	Date	Estimated		Actual			To date				
		Time	Units	Time	Units	Rate	Time	Units	Rate	Max	Min
1	2/9	120	60	168	87	0.52	168	87	0.52	0.52	0.52
Description: write program 1 (minutes per program)											
... (other activities types, so omitted on slide)											
3	2/15	205	148	245	133	0.54	413	220	0.53	0.53	0.52
Description: write prog. 2											
...											
6	2/24	150	87	145	59	0.40	558	279	0.50	0.53	0.40
Description: write program 3											
10	3/6	175	95	230	104	0.45	-----	-----	-----	-----	-----
Description: write prog. 4											

Quick Survey Results

- What's best prog. language?
 - a. C++ 24
 - b. Java 1
 - c. Perl
 - d. Visual Basic 2
 - e. Other 1
- You prefer to use:
 - a. C++ 24
 - b. Java 1
 - c. Other 3
- Your largest program:
 - a. < 500 loc 1
 - b. > 500 & < 1 kloc 10
 - c. > 1 & < 10 kloc 11
 - d. > 10 & < 100 kloc 5
 - e. > 100 kloc 1
- What's a kloc?
 - a. comments inc.? 5
 - b. declarations? 4
 - c. only exec. stmts? 14
 - d. number of CRs? 5

Mail to cmo@cs.odu.edu

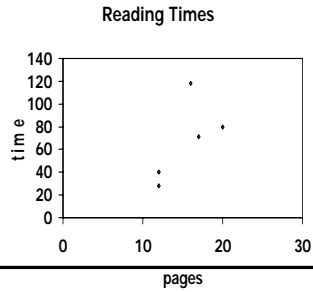
Comments

- After a while, you will have data on programming
 - avg, min, max
 - (really need better units than "program")
- Time to read a text chapter
 - better units are probably pages rather than chapters
- And other activities
- This can help with time estimates for future similar tasks

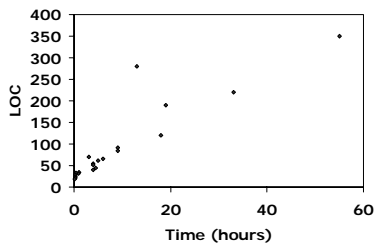
Units for reading

- Table 6.1 has student data on minutes/page.
 - Note variance
 - minutes/page varies from 3.3 to 7.4 for this student
 - Still, can be helpful to distinguish between time to read large chapter and time to read small.

Plot of reading times



Plot of programming time, C++ Objects



Units for programming

- LOC looks like a good predictor of time
 - It is. Not perfect, but lots of studies show a strong correlation between LOC and development time
- The problem is we only know LOC when we're done. We're looking for something to predict the time required for a task.
- So don't know LOC when we need them
- A solution is discussed later

What's an LOC?

- Depends on formatting
 - See text for examples
- Depends on what you count
 - Comments?
 - Only executable lines?
 - Compiler directives?

Basic metrics

- For our purposes LOC:
 - As long as you are consistent, it will work
 - For CS 350, use "grep ";" *.h *.cpp | wc -l"
 - Why? it's easy and good enough. then we all do it the same way.
- Basic productivity measure is LOC/hr
 - Usually misleading to compare your numbers and my numbers
 - Count loc differently
 - Likely depends on problem domain
 - Depends on what's included in your hour vs mine:
 - Testing, Documentation, Code/design/test plan reviews (etc.)?

Estimating Background

- Estimating models in other fields
 - Large base of history
 - In wide use
 - Generate detailed planning data
 - Require a size estimate as input
- Software size estimating experience
 - 100%+ errors are normal
 - Few developers make estimates
 - Fewer still use orderly methods

Size Estimating Principles - 1

- Estimating is an uncertain process.
 - No one knows how big the product will be
 - The earlier the estimate, the less is known
 - Estimates can be biased by business and other pressures (sometimes called "gutless" estimations)
- Estimating is an intuitive learning process.
 - Ability improves with experience
 - Some people are better at estimating

Planning Forms, assignment 1

- Plan/size est.
 - How big (LOC)
 - How long (minutes)
- Base this on your analysis of requirements
 - And your experience in writing other programs
 - And your impression of how long those took

Defect Record Log (pg 47) - 1

- Assumptions: After writing your code, you type it in, your proofread your typing
 - You may do assignment in steps, so this may be all of the program or only part.
- **Any** change made after proofing get recorded on form
- If you rely on compiler to proofread, then every thing it finds that you must change appears on form

Defects - 2

- Date
- Number: count them. start with "1", then "2", etc.
- Type: ignore for now
- Inject: in what phase did you introduce the mistake
- Remove: in what phase did you find the mistake
- Fix time: how long did it take to fix mistake
- Fix defect: ignore for now

Time Recording Log

- Already discussed

Test Plan

- Test case number: count starting with 1
- Test objective: short description of what this test case is checking for
- Testing procedure: how to run test
 - Often as simple as:
 - type "prog1 parm1 parm2 parm3 < test.case.1"
 - Sometimes instructions are needed
 - If, for example, the objective of the test is to see that the program responds appropriately to error in command line parameters
- Name of test file, if any.
- Expected output: tell tester what a correct version of the program will produce if this test is run.

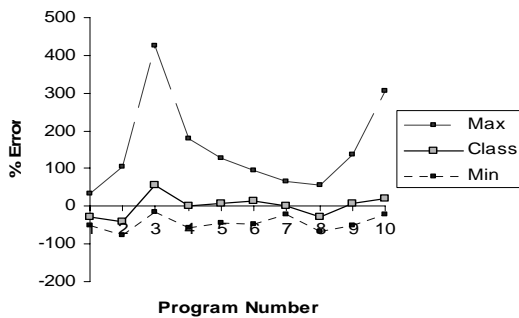
Project Plan Summary

- How much time was spent in each phase (from time log)?
 - Planning, Code design, Test design, Coding, Compiling, Testing, Postmortem
 - Total time spent
- How big did it turn out to be (from LOC counter)?
- How many defects were inserted in each phase
- How many defects were found in each phase
 - Both from defect log

Size Estimating Principles - 2

- Estimating is a skill.
 - Improvement will be gradual
 - You may never get very good
- The objective, however, is to get consistent -- at least "unbiased."
 - You will then understand the variability of your estimates
 - You seek an even balance between under and over estimates

Size Estimating Errors - 12 Students



Size Estimating Principles - 3

- The principal advantages of using a defined estimating method are:
 - You have known practices that you can work to improve.
 - It provides a framework for gathering estimating data.
 - By using consistent methods and historical data, your estimates will get more consistent.

Estimating Approaches

- Delphi
- Fuzzy logic
- Also mention Function Points since they are widely used

Delphi Size Estimating

- Uses several people to make estimate
 - Each makes an independent estimate
 - Each submits estimate to a coordinator
- Coordinator: returns to each estimator
 - Calculates average estimate
 - Enters on form: average, other estimates (anonymous), and previous estimate
 - That person's previous estimate (if any)
- Repeat. When re-estimates stabilize
 - Average is the estimate
 - Range is range of original estimates

Delphi Example - 1

- 3 estimators are asked to estimate the product.
- Their initial estimates are:
 - A - 13,800 LOC
 - B - 15,700 LOC
 - C - 21,000 LOC
- The coordinator then
 - Calculates average estimate as 16,833 LOC
 - Returns this with their original estimates to the estimators

Delphi Example - 2

- The estimators then meet and discuss the estimates.
- Their second estimates are
 - A - 18,500 LOC
 - B - 19,500 LOC
 - C - 20,000 LOC
- The coordinator then
 - Calculates average estimate as 19,333 LOC
 - Asks the estimators if they agree with this as the estimate

Delphi Size Estimating - 2

- Advantages
 - Can produce very accurate results
 - Utilizes organization's skills
 - Can work for any sized product
- Disadvantages
 - Relies on a few experts
 - Is time consuming
 - Is subject to common biases

Fuzzy Logic Size Estimating - 1

- Gather size data on previously developed programs
- Subdivide data into 5 size categories:
 - Very large, large, medium, small, very small
 - Establish size ranges
 - Include all existing and expected products
- Subdivide each range into subcategories

Fuzzy Logic Size Estimating - 2

- Allocate the available data to the categories.
- Establish subcategory size ranges.
- When estimating a new program, judge which category and subcategory it most closely resembles.

A Fuzzy Logic Example - 1

- You have historical data on 5 programs as follows:
 - a file utility of 1,844 LOC
 - a file management program of 5,834 LOC
 - a personnel record keeping program of 6,845 LOC
 - a report generating package of 18,386 LOC
 - an inventory management program of 25,943 LOC

A Fuzzy Logic Example - 2

- You establish 5 ranges, as follows
 - $\log(1844) = 3.266$
 - $\log(25,943) = 4.414$
 - the difference is 1.148
 - 1/4th this difference is 0.287
 - the logs of the five ranges are thus spaced 0.287 apart
 - the limits of these ranges are at 0.1435 above and below the midpoint of each range

A Fuzzy Logic Example - 3

- The 5 size ranges are thus:
 - very small - 1,325 to 2,566: file utility
 - small - 2,566 to 4970: no members
 - medium - 4,970 to 9,626: file management and personnel record program
 - large - 9,626 to 18,641: report generator
 - very large - 18,641 to 36,104: inventory management

A Fuzzy Logic Example - 4

- Your new program has the following requirements:
 - analyze marketing performance by product line
 - project the likely sales in each product category
 - allocate these sales to marketing regions and time periods
 - produce a monthly report of these projections and the actual results

A Fuzzy Logic Example - 5

- In comparing the new program to the historical data you make the following judgments:
 - substantially more complex application than the file management and personnel programs
 - not as complex as the inventory management program.
 - appears to have significantly more function than the report package.
- You conclude that the new program is in the lower end of "very large," or from 18 to 25 KLOC.

One more thing

- Experience shows that different kinds of code have different "typical" sizes
- So identify "generic" code types (see next slide)
- Use history data to compute typical sizes for each code type
- This is done once. Then used (until revised) for all new projects

To estimate size of new project

- For each component, identify generic type it most resembles
- Then for each component, compare it to a similar component from a previous project if possible
- Decide whether each is very small, small, medium, large or very large
- Sum individual size estimates to get estimate of project size

Fuzzy Logic Size Estimating - Advantages

- Fuzzy logic estimating
 - Is based on relevant historical data
 - Is easy to use (after you have the data)
 - Requires no special tools or training
 - Provides reasonably good estimates where new work is like prior experience

Fuzzy Logic Size Estimating - Disadvantages

- The disadvantages of fuzzy logic are
 - It requires a lot of data
 - The estimators must be familiar with the historically developed programs
 - It only provides a crude sizing
 - It is not useful for new program types
 - It is not useful for programs much larger or smaller than the historical data

Fuzzy Logic Sizes for C++, LOC per method

Category	Very Small	Small	Medium	Large	Very Large
Calculation	2.34	5.13	11.25	24.66	54.04
Data	2.60	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
Logic	7.55	10.98	15.98	23.25	33.83
Set-up	3.88	5.04	6.56	8.53	11.09
Text	3.75	8.00	17.07	36.41	77.66

From Humphrey, "A Discipline for Software Engineering," AW, 1995, pg. 117

Another widely used time estimation method

• Function Points

- Widely used in industry, particularly data processing/business applications
- Based on idea that the amount of “functionality” in a software product determines its development costs.
- IFPUG: an organization supporting training in use of function points

So back to units for programs

- Your units should be methods (member functions in C++)
 - So after coding, count the number of public and private functions in each object
 - These are the units for job number log
- Your predictions should be based on:
 - How you write code
 - How much time you take/LOC
- But to do this, you need data!

What did I not tell you?

1. Requirements **ALWAYS** change (need a requirements management process!)
2. Systems exist to solve problems, but any complex system changes the nature of the problem it was intended to solve.
3. Schedules must be sensitive to more issues than discussed here. But this type of information should be one factor.

Things to Remember about Size & Time Estimates

- Accurate size estimates will help you to make better development plans.
- Size estimating skill improves with practice.
- A defined and measured process provides a repeatable basis for improvement.
- There are several ways to make size estimates.
