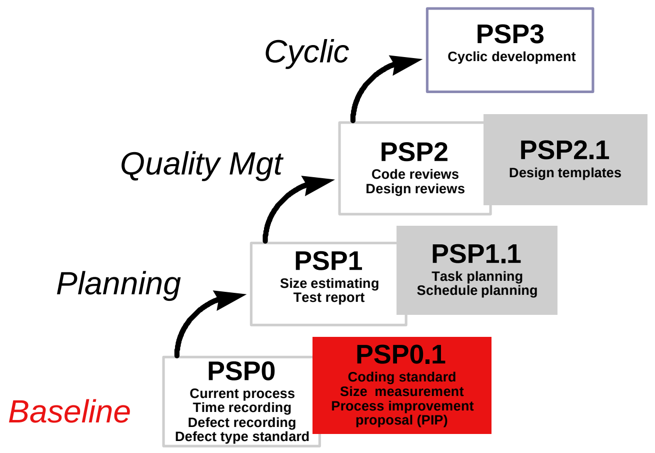
Planning II: Measuring Software Size

Outline

- Review of the PSP Levels
- Reasons for Measuring SW Size
- Criteria for SW Size Measures
- A Size Measurement Framework
- Using LOC Counts
- Reuse Considerations
- LOC Accounting
- Calculating Productivity
- LOC Counters
- Homework #2

Review of PSP Levels *(Humphrey, 1995, p. 11)*



Reasons for Measuring SW Size

(cf. Humphrey, 1995, p. 69)

- SW planning starts with estimating job size.
- By estimating the size of the product you plan to build, you can better judge the amount of work required to build it.
- In order to estimate you need historical data.
- Thus you should measure current projects in order to create a historical database.

Criteria for SW Size Measures

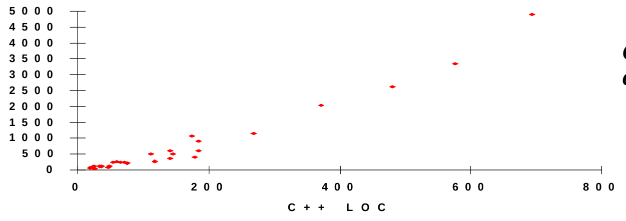
(cf. Humphrey, 1995, p. 69-74)

- Useful for Planning
  - *Is the measured information related to the information which we desire to predict?*
  - *Is it suitable for early planning? Can you visualize it at early stages of development? Can you measure a surrogate or comparable proxy early in, and throughout, development?*
- Precise (vs. Accurate / Reliable)
  - *Is the level of granularity of the measure appropriate with respect to the overall project?*
- Automatically Countable
  - *Can we directly count the measure automatically in the produced product?*

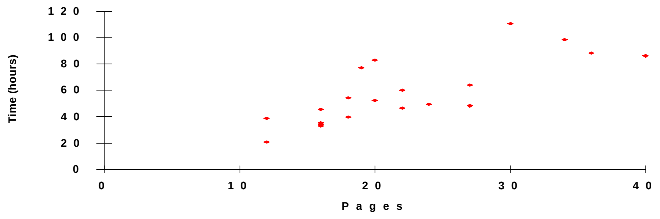
Usefulness: Correlation (r)

- *Correlation is a measure of how much two sets of data are related. CAUTION: "Correlation does not imply causation."*
- *r is the symbol for correlation.*
- *r varies from -1.0 to +1.0.*
- *+1.0 indicates a perfect positive relationship,*
  -1.0 indicates a perfect inverse relationship,
  0 indicates no relationship.
- *Ex: If age and height were perfectly related (r=1.0), knowing one's age would allow exact knowledge of one's height, with height increasing as age increases. However, since age and height are not perfectly related, knowing age allows prediction of height with an associated amount variation.*
- *If time spent on technical reviews were inversely related to number of defects reported, with, say r=-0.75, then knowing the amount of time spent on reviews would allow us to predict relatively accurately the number of defects that would be reported in the delivered product.*

Correlation Examples



Quite good correlation



Not as good correlation

Usefulness: Variance Explained (r^2)

- r^2 tells how much of the variation in one set of data is explained by the variation in the other set.
- r^2 is called “variance accounted for”, or VAF.
- For a correlation to be practically useful, r^2 should be > 0.5.

Usefulness: Significance (a)

- *A correlation's a (alpha) indicates the statistical significance of the correlation, or the degree to which the correlation would be expected simply due to chance.*
- *The smaller the a the less likely the results are simply due to chance.*
- *0.05 is generally accepted as a "good" a level; sometimes even up to 0.1 is accepted.*
- *Ex: If data are collected 100 times and we determine r=0.75 with =0.05, then we would expect to obtain a correlation of 0.75 5 times simply because of random chance. If r=0.75 occurs more than 5 times in 100 then we might conclude that there is something other than random fluctuation which is causing it to occur.*

Usefulness: Suitable for Early Planning

- Can you visualize it at early stages of development?
  - Ex: Square feet in a house, vs. number and type of rooms.
- Can you measure a surrogate or comparable proxy early on, and throughout, development?
  - Ex: PC (permitted collaborations), PI (permitted interactions), & Objects in object-oriented analysis, design, and coding.

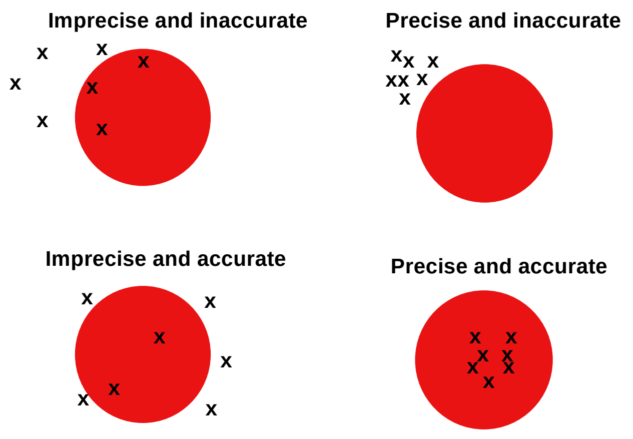
Precision vs. Accuracy & Reliability

(cf. Humphrey, 1995, p. 67, 69, 73)

- *Humphrey seems to confuse these terms and use them in different ways at different times.*
- *Precision*
  - *Precision = the granularity of the measure with respect to the overall project. The level of granularity should be appropriate for the project.*
  - *Ex: appointment at 8:16am vs. around 8am*
  - *Ex: test time of 2 minutes vs. 1/4 day out of 2-day project*
  - *Ex: on the 25th vs. during the 4th week of a month-long project*
- *Accuracy & Reliability*
  - *Accuracy = the relation between an assertion and an actual fact.*
  - *Reliability = the repeatability of a measure over people and projects.*
  - *Ex: Estimate 5 hrs, actual 7 hrs. Accuracy = -2 hrs (underestimated)*
  - *Ex: Estimate 5 hrs, actual 4 hrs, Accuracy = +1 hrs (overestimated)*
  - *Ex: Developers 1, 2, and 3 all come within 10% of their estimate using a given measure and process.*

Precision vs. Accuracy

(from Humphrey, 1995, lecture slides)



Note: Humphrey is using the term “precision” here in the sense of “reliability”, that of obtaining the same results over and over, not in the sense of “granularity”.

Requirements for Precision, Reliability, and Accuracy

- Precision requires knowing the big picture.
- Reliability requires well-defined measures. Coding and LOC counting standards facilitate this.
- Accuracy requires a relevant collection of historical data and a method for extrapolating from it.

Automatically Countable

(cf. Humphrey, 1995, p. 74)

- Reasons for automated LOC counting:
  - Manual counting is:
    - tedious
    - time-consuming
    - error-prone
    - practically impossible for large projects
  - Automated counting provides:
    - accurate results
    - economic use of resources

Two Criteria in the SEI's Size Measurement Framework

(cf. Humphrey, 1995, p. 74)

- Communication
  - *"Will others know [exactly] what has been measured…?"*
  - *Precision*
- Repeatability
  - *"Would someone else be able to repeat the measure and get the same result?"*
  - *Reliability*

LOC Counting Standard

(cf. Humphrey, 1995, p. 74-78)

- *The Counting Standard includes:*
  - *Definition, Author, Date*
  - *Language*
  - *Count Type*
    - *logical vs. physical*
  - *Statement Type*
    - *executable & non-executable*
  - *Clarifications*
- *cf. Table 4.1 (LOC counting template) on p. 74*

Logical vs. Physical LOC
(cf. Humphrey, 1995, lecture slides)

- Logical LOC
  - *Independent of format & editing changes*
  - *Correlate with development effort*
  - *Uniquely definable*
  - *Complex to count*
- Physical LOC
  - *Easy to count*
  - *Not independent of format & editing changes*
  - *Not uniquely definable*

The PSP Counting Standard
(cf. Humphrey, 1995, lecture notes, and p. 92)

- *Uses a coding standard and physical LOC counter.*
- *Thus, 1 logical line = 1 physical line.*
- *The coding standard must be followed faithfully.*
- *Count all statements:*
  - *begin, end, if, then, else, etc.*
  - *{, }, ;, ., etc.*
  - *count declarations, directives, headers, etc.*
- *Do not count blanks, comment lines, automatically generated code, or reused code.*
- *Count new and changed code for measuring and estimating development productivity.*
- ***"In no case should you compromise program function or readability to simplify LOC counting."***

Counting Example

(based on Humphrey, 1995, lecture notes example and Dan Turk's counting standard)

```
// I_set_set() determines if n is in an "integer set" collection.
// It returns TRUE if so, and FALSE if not.
void iset_set (
    int *n;
    bool_t inc; ) {

    inc = FALSE;
    search_ptr = set_start;
    while (search_ptr != NULL && inc == FALSE) {
          if (searth_ptr->this_n == *n) then
                    inc = TRUE;
          else
                    search_ptr = search_ptr->next_n;

    } // iset_set()

// In this example, LOC = 11.
```

LOC Counting: The Big Picture

(cf. Humphrey, 1995, p. 75, 78, & lecture slides)

- Many decisions will need to be made when defining a LOC Counting Standard.
- In order to make these decisions you need to have a big picture, or general approach, that guides you.
- Humphrey's general approach:
  - Count logical statements
    - Count all semicolons and selected keywords.
    - Logical statements measure content rather than format, which physical lines measure...
  - Omit blank lines and comments.
    - A separate coding standard addresses good coding practices such as whitespace and comments.
  - Count and record each language separately.

LOC Counting Standard & Examples

- Look at Tables 4.1 (template), 4.2 (Pascal example), & 4.3 (C++ example) on p. 74-78

Using LOC Counts

- As is well known, LOC counts can be easily misinterpreted and misused.
- Don’t mix LOC counts from different languages and types of code (i.e. test, support, product, …)
- Use appropriate measures of different attributes of a program.
  - Packaging
  - Evaluating development work
  - Assessing program quality

Using LOC: Packaging

(cf. Humphrey, 1995, p. 81-82)

- Physical product shipping volume, execution and disk storage memory requirements are very important packaging issues.
- Many factors are important here:
  - Documentation size (including comments, etc.)
  - Size of executable code (vs. source)
- Not discussed further in this course.

Using LOC: Evaluating Development Work (cf. Humphrey, 1995, p. 81-82)

- Use LOC count numbers which seem most appropriate to the task at hand. The correct answer will vary from task to task.
- Ex: When evaluating development status (where are we?), new and changed LOC is probably most relevant.
- Ex: When forecasting a new product, don't consider unmodified reused code.

Using LOC: Assessing Quality

(cf. Humphrey, 1995, p. 83-84)

- *For our purposes, defect counts are a surrogate for program quality, even though there are other issues (e.g. whitespace, comments, etc.).*
- *Most frequent measure = new & modifies LOC.*
- *However, when comparing two completed products, total LOC may be the best predictor of future maintenance effort.*
- *When using total LOC in assessing quality, small changes to large programs appear insignificant. However, defects per changed LOC may be nearly 40x more likely than that on new code. Thus using defects per new and changed LOC is probably the best measure.*

Reuse Considerations

(cf. Humphrey, 1995, p. 84-85)

- *Reuse is one of few approaches that promises potentially orders of magnitude of improved productivity.*
- *However, it is difficult to motivate reuse.*
  - *Developer measurements (based primarily on new and modified code) look "worse".*
  - *Development database gets "corrupted" by new, modified, and reused components, and thus it becomes harder to forecast.*
- *There are almost an infinite number of reuse types.*
  - *Copy sections of code*
  - *Modify existing code*
  - *Inheritance*
  - *Function libraries*
  - *Etc.*

LOC Accounting (cf. Humphrey, 1995, p. 85-89)

- *Various anomalies may arise from counting the LOC in a program at various points in its development history.*
- *cf. Example*
  - *25 less LOC than expected, p. 86-87*
  - *correct analysis, p. 88*
    - *base v0 = initial added*
    - *base v1 = base v0 + modified + added + reused - deleted - modified*

Calculating Productivity
(cf. Humphrey, 1995, p. 88-90)

- *WARNING: You can generate almost any productivity number you want by changing the way you count LOC and calculate productivity.*
- *Using new and changed LOC is usually most appropriate for new development, and should be used for all productivity calculations in this course.*
- *Other methods may be more appropriate for maintenance.*
- *Later, after collecting a historical database you can then adjust your calculations as you best determine.*

LOC Counters (cf. Humphrey, 1995, p. 90-94)

- Physical
- Logical
- Physical + Coding Standard
  - Must strictly follow the coding standard.
  - Can modify counter to address issues such as counting { and } as 2 LOC...
  - ***"In no case should you compromise program function or readability to simplify LOC counting."***
- Counting Deletions and Modifications
  - Quite difficult problem.
  - Can do manually for small programs.
  - Can include code in LOC counter (or create a separate "diff" program) to determine this from versions N and N-1.

Homework #2

- Reports R1 & R2
  - LOC counting standard
  - Coding standard
  - See p. 767-769, and p. 76 template
- Program 2A
  - Program LOC counter
  - Reports R1 & R2 must be completed first.
  - See p. 753, and Assignment Kit #2
- PIP
  - Process Improvement Proposal
  - To be completed for all programs from this point on.
  - See p. 668, 669, and Assignment Kit #2