Disciplined Software Engineering
Lecture #2

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Lecture #2 Overview
- Planning I and II
Planning overview

Software size
• why measure size?
• size measurement criteria
• the SEI size measurement framework

Counting program size
• counters
• coding standards
Why Make Plans?

To make commitments you can meet

To provide a basis for agreeing on the job

To guide your work

To help track your progress

To project completion

The Project Planning Framework
Why Measure Size?

To make better plans

To assist in tracking development

To normalize other measures
  • development resources
  • defect rates

Size Measurement Criteria

Relationship to development effort

Precision

Machine countable

Suitable for early planning
Size Versus Development Effort

The principle requirement: if the size measure is not directly related to development cost, it is not worth using.

There are many possible measures:
• lines of code (LOC)
• function points
• pages, screens, scripts, reports

The size measure should be sensitive to language, design, and development practice.

C++ LOC Versus Development Time

![C++ LOC vs Development Time Graph]
Pascal LOC Versus Time

Text Pages Versus Time
Screen LOC Versus Time

Relationship to Development

LOC is a reasonably good measure for development of source programs like Pascal and C++.  

Pages are an acceptable measure for document development.  

LOC is not an adequate measure for screens, reports, or scripts.  

Some other possible measures are function points, screens, and modules.
**Precision and Accuracy**

Imprecise and inaccurate

Precise and inaccurate

Imprecise and accurate

Precise and accurate

---

**Measurement Precision**

When 2 people measure the same thing will they get the same result?

To do so requires a precise measurement definition.

The measure must also be properly applied.

- Pascal LOC do not equate to assembler LOC
- new LOC do not equate to modified LOC
- logical LOC do not equate to physical LOC
- C++ LOC may not relate to C++ LOC
Machine Countable

Size measurement is time consuming and inaccurate.

Automated counters can only work on definable program characteristics.

Counters can be complex:
• size definition selected
• counting method

Suitable for Early Planning - 1

The issue is: what can you visualize early?
• for a house, square feet predict cost
• few people can visualize a house in terms of square feet of living space
• numbers of rooms are more intuitive

Needed for good plans:
• intuitive size measures
Suitable for Early Planning - 2

Unfortunately, the popular intuitive measure is not measurable and the popular measurable measure is not intuitive.

Function points
• intuitive
• not directly measurable

LOC
• not intuitive
• directly measurable

Selecting a Size Measure - 1

Start with product development data
• resources required
• product characteristics
• any special development conditions

Rank products by resources required.

See what characteristics distinguish those products that took the greatest effort from those that took the least.
Selecting a Size Measure - 2

See if these differences are measurable.
• correlate this measure for the product set
• if no correlation, try again

There may be no single best measure.
• a combination of measures could be needed
• methods for handling multiple measures are discussed later

Selecting a Size Measure - 3

If you are better at estimating resources than program size, size estimation will not improve your planning.

If you estimate resources directly, you must:
• keep accurate records
• build a large database
• use an estimating guru
The SEI Measurement Framework

Logical versus physical lines

Statement specifications:
• executable
• nonexecutable
• counted statement types

Application:
• language and code type
• origin and usage

Counting Program Size - 1

Logical lines
• invariant to editing changes
• correlate with development effort
• uniquely definable
• complex to count

Physical lines
• easy to count
• not invariant
• not uniquely definable
Counting Program Size - 2

The PSP uses a coding standard and a physical LOC counter.

• uses a physical line for each logical line
• uses a defined coding standard
• this standard must be faithfully followed

Then physical line counting equals logical line counting

A Counting Example

procedure ISet.Set(var N: int; var inc: boolean);
begin
  inc := false;
  SearchPtr := SetStart;
  while (SearchPtr<>nil) and (inc == false) do
    if SearchPtr^.ThisN == N
      then
        inc := true
      else
        SearchPtr:=SearchPtr^.NextN;
  end;
The PSP Counting Standard

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.

Line of Code Accounting

For small programs, size tracking can be done manually, but it requires care.

For larger programs, size tracking requires an accounting system.

LOC accounting provides an orderly and precise way to track LOC changes through multiple program versions.
Example of LOC Accounting - 1

- Version 0
  350 LOC

- Enhance to Version 1
  + 125 New and Changed LOC

- Expected Size:
  350 + 125 = 475 LOC

- Measured Size: 450 LOC

What happened?

Example of LOC Accounting - 2

<table>
<thead>
<tr>
<th>Added</th>
<th>Subtracted</th>
<th>Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base V0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Deleted</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Modified</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Added</td>
<td>350</td>
<td></td>
</tr>
<tr>
<td>Total V0 LOC</td>
<td>350</td>
<td>-0</td>
</tr>
<tr>
<td>Deleted</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Modified</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Added</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Final Product</td>
<td>125</td>
<td>-25</td>
</tr>
<tr>
<td>Total New and Changed LOC</td>
<td>475</td>
<td></td>
</tr>
</tbody>
</table>
PSP0.1 Process Additions

Coding and LOC counting standards:
• tailored to your language and needs
• incorporate PSP elements from the text
• needed to write the PSP programs

Process improvement proposal (PIP)
• used to record process improvement ideas
• also used to record findings and comments on each project

PSP0.1 Summary Additions - 1

Program size - before development:
• if this is an enhancement, measure the size of the base program
• enter these LOC in the Base (B) space under Actual
• estimate the new and changed LOC
• enter these LOC in the Total New & Changed (N) space under Plan
PSP0.1 Summary Additions - 2

Program size - after development:
• measure total program size
• enter these LOC in the Total LOC (T) space under Actual
• count the deleted LOC and enter in the Deleted (D) space under Actual
• count the modified LOC and enter in the Modified (M) space under Actual
• count the reused LOC and enter in the Reused (R) space under Actual

PSP0.1 Summary Additions - 3

After development (continued):
• calculate the added LOC as:
  \[ A = T - B + D - R \]
• enter the added LOC in the Added (A) space under actual
• count or estimate the number of new and changed LOC that will be added to the reuse library
• enter as Total New Reuse in the space under Actual
PSP0.1 Summary Additions - 4

To Date:
• total the actual reused, new and changed, total, and total new reuse LOC from this and all previous programs
• enter these data in the To Date column for each LOC type

These To Date data are used to calculate various process parameters in later PSP versions.

Completing the PSP Form

In this course, you are asked to complete a PIP form for each programming exercise and include the following information:
• Problem description - any problems you have encountered in using the process
• Proposal description - any suggestions you have for process improvements
• Notes and comments - your observations and findings from doing the exercise
Assignment #2

Read chapters 3 and 4 of the text.

Produce an LOC counting standard and a coding standard.

Use PSP0.1 to write program 2A to count the logical LOC in your programs. Use a coding standard and a physical line counter.

See Appendices C and D for process and program specifications.

Messages to Remember from Lecture 2 - 1

1 - To effectively plan and manage your work, you must measure product size.

2 - The PSP uses LOC as the size measure.
Messages to Remember from Lecture 2 - 2

3 - For other measures, size must correlate with development time.

4 - Every size measure should be precisely defined and automatically countable.