Instructor Notes for L03

Personal Software Process for Engineers

# Recommended Order of Presentation

1. Account for Size Categories (PowerPoint)
2. WS\_E\_03 (PowerPoint) with handout ASGKIT\_E\_03\_Module\_Counter (Word)

## Account for Size Categories

One of the “back to the future” changes in this sequence is the separate introduction of post-development size accounting during evaluation and pre-development size accounting during planning. This process step that introduced post-development accounting used to be known only as PSP0.1, and it is still referred to as “your current process” in WS\_E\_03 since we lacked a better name.

Whatever changes the instructor/learners made to the process script (INITIAL+1?) should be further enhanced to take into account the necessary count and capture of (B)ase code that is the starting point (which can obviously be done during planning, and even be known prior to planning), (D)eleted code, and (M)odified code. (This means that (U)nmodified = B − D − M.) This approach should then lead to an INITIAL+2 script. The instructor/learners should feel free to create more meaningful names.

The general philosophy of the course is to keep it short, simple, and to the point; hence no long discussions of the math are involved. Introduce the concept, introduce the diagram, show the tabular and diagram example, and get on with the exercise.

Experienced PSP instructors will note the absence of the (R)eused and (NR) New Reused terms. Arithmetically there is no difference between (R)eused code and (B)ase code that has not been touched (i.e., “unmodified”), and (R) and (NR) have consistently added confusion to PSP terminology without providing much value to developers. This version of PSP deliberately eliminates these terms, which seems also to eliminate a lot of potential confusion and not necessarily productive classroom discussion.

## WS\_E\_03

This exercise is to enhance the simple physical LOC counter from the previous lesson so that it can keep track of the size of the internal parts of a module—such as classes, functions, subroutines, and the like—along with the overall total size of a module. The intent is to expose learners to counting concepts using their own code. The accompanying Word file gives learners an exemplar for useful and usable output formatting.

Document Markings

Copyright 2020 Carnegie Mellon University. All rights reserved.  
  
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.  
  
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.  
  
NO WARRANTY. THIS MATERIAL IS FURNISHED ON AN “AS-IS” BASIS WITH NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, ANY WARRANTY WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT, OR THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.  
  
[Distribution Statement A] This material has been approved for public release and unlimited distribution. The United States Government has Unlimited Rights in this material as defined by DFARS 252.227-7013.

The text and illustrations in this material are licensed by Carnegie Mellon University under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

The Creative Commons license does not extend to logos, trade marks, or service marks of Carnegie Mellon University.