Software Measurement for Science and Profit

Robert C. Leif and Suzanne B. Leif
Ada_Med, a Division of Newport Instruments
E-mail rleif@rleif.com
(619) 582-0437

1. Introduction:

Experimental scientific knowledge is often the result of quantitative measurements. The technology originally employed for these measurements subsequently can be employed in commerce.

```
Balances => Scales
Trajectory calculations => Accounting Software
```

Quantitative measures of the amount of Software are needed to

a) Measure efficiency.

- **The amount of product produced divided by the effort.**

b) Measure contributions to a project.

- **For purposes of estimation and compensation**

2. Software Measurement Units

2.1 Lines of Source Text (Lines)

```
Lines := Total_Semicolons - Comment_Semicolons;
```

- **Inappropriate for a language with any means of reuse:**
  subroutines, generics, classes, etc.

2.2 Function Points

Function points were originally described by Albrecht & now are described by the International Function Point Users Group's (IFPUG) Release 3.0 of the Function Point Counting Practices Manual, 1990.

From What Are Function Points? By Capers Jones, Chairman, Software Productivity Research, Inc.

http://www.spr.com/library/0funcmet.htm

```
Function_Points := 4*Inputs + 5*Outputs + 4*Inquiries +
  10*Data_Files + 7*Interfaces;
Complexity_Adjustment : Finagler_Factor _Type;
    --often required
```

2.3 Feature Points

Capers Jones extended Function_Points to:

```
Complexity : positive range 1..10 :=3;
Feature_Points := Function_Points  + Complexity*ALGORITHMS
  - 3*Data_Files;
```

2.4 Corrected Lines of Source Text

Ada 95 and other OO languages probably require a minor change to Lines of Source Text (Corrected _Lines).

```
Corrected_Lines := Total_Semicolons
  - Comment_Semicolons
  - 0.75*Renaming_Semicolons
  --contained in declarations.
  - 0.75*Subtype_Rename_Semicolons;
  --Subtype declarations which
  --do not include a range.
```

- **Could this be done with a tool based on ASIS?**

2.5 . Linked Lines of Source Text (Linked_Lines)

equals the total number of semicolons which would be actually used in the linked Executable if

a) all loop structures remained intact (no unrolling);

b) all instantiations of generics are treated as the equivalent of the source text which would have been created without the use of the generic;

c) all instances of inherited subprograms of tagged types are treated as the equivalent of the source text which would have been created without the use of the tagged type.

- **Could this be done with a tool based on ASIS?**

2.6 Source Text Efficiency (Source _Efficiency) is a measure of the product produced versus the effort to create the software.

```
Source_Efficiency:= Linked_Lines/Lines_Source;
```