Automated Collection of Software Sizing Data

Briefing at the
University of Southern California - Center for Software Engineering
USC - CSE
Annual Research Review

March, 1997

George E. Kalb
Advanced Software Technology & Business Development
Electronic Sensors and Systems Division
Northrop Grumman Corporation
Post Office Box 746 - MS 432
Baltimore, Maryland 21203
Telephone 410-765-8727
George E. Kalb

Mr. Kalb is a Fellow Engineer with over fourteen years at Northrop Grumman ESSD and is responsible for advanced software technologies and business development activities. His engineering experience includes Principal Investigator for the Avionics Fault Tolerant Software (AFTS) Program and the associated Ada Technology Insertion Program (ATIP) and has been involved with numerous CR&D contracts at WL/AAAF-3. His management experience includes the highly successful Generic DisAssembler Program and numerous Proposal Preparation activities where on-time, on-budget performance is critical. Past engineering activities include AN/APG-68 Operational Flight Program (OFP) software development, J-STARS & SDI Ground Base Radar proposal signal processing software architect, YF-23 & YF-22 processor timing and sizing studies, and lead software engineer for the E-3A Programmable VHSIC Signal Processor Operational Software development.

Mr. Kalb is responsible for conceiving, developing, and successfully distributing a set of source lines of code counting tools that are the foundation for ESSD’s software metrics program.

Mr. Kalb is also a faculty member at the G.W.C. Whiting School of Engineering of the Johns Hopkins University since 1988 where he teaches a graduate-level course in Software Engineering and the Software Lifecycle and was the recipient of the 1995 Johns Hopkins University “Excellence in Teaching” award.
THE MANY USES OF SIZING DATA

- Software Cost Models
  - cost estimation
  - project post-mortem
- Metrics Programs
  - historical project data
  - productivity
  - defect rate
- Development Projects
  - amount of reuse software
  - estimated / actuals overall project size
- Software Quality Assurance
  - amount of commentation in source code product
  - use of language constructs
  - defect rate calculations
- Re-Engineering Activities
  - language translation
  - platform translation
- Compiler Efficiency Analysis
WHAT TO COUNT?

**LANGUAGE-LEVEL**
- Deliverable Source Instructions
- Machine Instructions
- Semicolons
- Logical Source Statements
- Executable Lines
- Data Declarations
- Comments
- Compiler Directives

**PROJECT-LEVEL**
- Include Files
- Generics
- Reuse Code
- Unused Code, Dead Code
- Deleted Code
- Support Software
- Patch Files, Data Files
- Test Software, Command Files
- Multiple Language Implementations
- Off-The-Shelf Software
- Non-Deliverable Software
- Modified Software

LOC Definitions & Language-Level Counting Rules

Project-Level Counting Rules
QUEST FOR AUTOMATION

- Automation of collection of software sizing data:
  - decreases subjectivity (estimates)
  - improves accuracy and consistency of information
  - reduction in time and effort required to gather data

- Solution: Let’s go make / buy a tool

  How will it be used?
  What will it count?
  What resources are available for its development?

  Who’s tool?
  What does it do?
  How can I use it?
  Suitability for multiple uses?
PROBLEMS WITH AUTOMATED LOC COUNTING TOOLS

- Developed to satisfy a short-term project specific need
  - no documentation
  - no visibility into LOC definition used
  - use of rule-of-thumb conversion factor(s)
  - incorrect counting
    - misclassification of language constructs
  - produced ambiguous output
  - developed using substandard practices
  - not validated or maintained
HOW TO BUILD A BETTER LOC COUNTING TOOL

• LOC definition utilized must:
  - span multiple programming languages
    » 3GL HOL (mixed language implementations)
    » assembly
    » microcode
    » test languages
  - be easily interpreted
  - provide for project utility
  - be compatible with software cost model use
  - be amenable to automation

• Deliverable Source Instruction (DSI) concept utilized
  - easy to understand & automate
  - may be applied consistently across programming languages
  - does not require subclassification of language constructs
  - incorporated into many software cost models
PROJECT-LEVEL COUNTING RULES

• What to count is often dependent on end-use of sizing data
  – count all source code
  – count all new source code
  – count reuse and off-the-shelf software
  – count deleted code
  – count support software

• The process of collecting sizing information should not be coupled to project-level concerns
  – concentrate on automation of language-level collection processes
  – project-level concerns addressed as a post process
Northrop Grumman Corporation ESSD LOC COUNTING TOOLSET

• Set of automated LOC counting tools (CodeCount™)
• Programming languages analyzed:
  – Ada (Mil-Std-1815A and Ada95)
  – FORTRAN
  – Jovial J73 (Mil-Std-1589B)
  – ANSI-C and C++
  – Mil-Std-1750A Assembler
  – Pascal
  – two proprietary microcode languages
• Share common operational requirements
  – set up of input files
  – operation / execution
  – information content & format of output produced
• Developed and successfully used for over a decade
• Satisfies variety of end-user requirements (p.#3)
• Basis of metrics data collection process
**CodeCount™ Installation Procedure**

**INITIALIZATION**

- Load *ada_lines.c* & *ada_lines_environment.dat* Files
- Edit & Set User-Defined Parameters *ada_lines_environment.dat*
- Compile & Link *ada_lines.c*
- Files contain inclusive documentation

**EXECUTION**

- Use Directory Command to Create Input File of Filenames *ada_list.dat*
- Execute *ada_lines.exe*
- Review Results *ada_outfile.dat*
CodeCount™ CONTEXT DIAGRAM

File of Filenames
*ada_list.dat*

File of User Defineable Data
*ada_lines_environment.dat*

File to be Analyzed
*<filename>.ada*

File of Results
*ada_outfile.dat*

SLOC COUNTING TOOL

Progress Indicator & Error Messages
“Ada_Lines_Environment.dat” File

**Temporary Project Name**

(* Project_Name, in 45 spaces *)

1 (* QA_Switch *)

0 (* Compare_Spec *)

132 (* Line_Length *)

60 (* Min_Percent *)

10.0 (* Inc_Percent *)

---

(* Refer to the source code file, ‘ada-lines.c’, for further information pertaining to the INSTALLATION PROCEDURES and EXECUTION PROCEDURES of this code counting tool. *)

(* Note: *)

(* 1. The above user-defined parameters must be spaced one entry per line of this file. Numeric entries, with the exception of ‘Inc_Percent’, are of type Integer. *)

(* 2. The ‘ada-lines-environment.dat’ file must be co-located in the directory/path whereas the code counting tool is to be invoked. Failure to do so will result in the insertion of predefined default values for the entries contained herein. *)

---

**USER DEFINEABLE PARAMETERS**

(*)

(* Project_Name -- Allows the user to insert the name of the Program or Project that the source code to be counted pertains. *)

(* The Project Name will appear within at the headings of the ‘ada_outfile.dat’ file produced upon execution of the ‘ada-lines’ code counting tool. *)

(* QA_Switch -- Allows the user to turn on ‘I’ or to turn off ’O’ the reporting of programming language reserved word usage as found in the summary page of ‘ada_outfile.dat’. *)

(* Compare_Spec -- Allows the user to control the case sensitivity of the code counting tool. A setting of ‘I’ indicates that full case sensitive comparisons must be made. A setting of ’O’ allows valid comparisons to occur between like letters of upper and lower case. *)

---

NORTHROP GRUMMAN
"Ada_List.dat" File

DISK1:[DABBAR.ADA] A.ADA;1
DISK1:[DABBAR.ADA] BANK.ADA;40
DISK1:[DABBAR.ADA] BANK2.ADA;54
DISK1:[DABBAR.ADA] CHANGE.ADA;9
DISK1:[DABBAR.ADA] DEFINE_REAL.ADA;7
DISK1:[DABBAR.ADA] ELEVATOR.ADA;78
DISK1:[DABBAR.ADA] HM4.ADA;16
DISK1:[DABBAR.ADA] HMS_.ADA;5
no file.ada
DISK1:[DABBAR.ADA] HMS_.ADA;2
“Ada_Outfile.dat”
Total Number of Deliverable Source Instructions (DSIs) 46
Total Number of Ada Statements (SLOC) 44

<table>
<thead>
<tr>
<th>Reserved Word</th>
<th>Total Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELAY</td>
<td>0</td>
</tr>
<tr>
<td>pragma</td>
<td>1</td>
</tr>
<tr>
<td>exception</td>
<td>8</td>
</tr>
<tr>
<td>function</td>
<td>24</td>
</tr>
<tr>
<td>package</td>
<td>16</td>
</tr>
<tr>
<td>private</td>
<td>0</td>
</tr>
<tr>
<td>abstract</td>
<td>0</td>
</tr>
<tr>
<td>new</td>
<td>8</td>
</tr>
<tr>
<td>reserved</td>
<td>40</td>
</tr>
<tr>
<td>protected</td>
<td>0</td>
</tr>
<tr>
<td>until</td>
<td>0</td>
</tr>
<tr>
<td>abort</td>
<td>0</td>
</tr>
<tr>
<td>begin</td>
<td>2</td>
</tr>
<tr>
<td>case</td>
<td>8</td>
</tr>
<tr>
<td>null</td>
<td>3</td>
</tr>
<tr>
<td>raise</td>
<td>2</td>
</tr>
<tr>
<td>return</td>
<td>15</td>
</tr>
<tr>
<td>task</td>
<td>6</td>
</tr>
<tr>
<td>terminate</td>
<td>0</td>
</tr>
<tr>
<td>access</td>
<td>1</td>
</tr>
<tr>
<td>declare</td>
<td>0</td>
</tr>
<tr>
<td>procedure</td>
<td>11</td>
</tr>
<tr>
<td>separate</td>
<td>0</td>
</tr>
<tr>
<td>aliased</td>
<td>0</td>
</tr>
<tr>
<td>toured</td>
<td>0</td>
</tr>
<tr>
<td>use</td>
<td>15</td>
</tr>
<tr>
<td>with</td>
<td>16</td>
</tr>
<tr>
<td>declare</td>
<td>0</td>
</tr>
<tr>
<td>accept</td>
<td>5</td>
</tr>
<tr>
<td>entry</td>
<td>0</td>
</tr>
<tr>
<td>goto</td>
<td>0</td>
</tr>
<tr>
<td>request</td>
<td>0</td>
</tr>
</tbody>
</table>

This output produced on Fri Mar 7 17:37:10 1997
REVISION AG13 SOURCE PROGRAM -> ADA_LINES
## "C_Outfile.dat" File

**C/C++ Source Lines of Code Counting Program**

(C) Copyright 1991, 1994 Northrop Grumman Corporation, CodeCount (TM)

Northrop Grumman Corporation retains ownership of this copy of software. It is licensed to you. Use, duplication, or sale of this product, except as described in the CodeCount License Agreement, is strictly prohibited. This License and your use of the software automatically terminate if you fail to comply with any provisions of the License Agreement. Violators may be prosecuted.

This product is intended to be used by Northrop Grumman Corporation (NGC)

### Project Name

**Temporary Project Name**

### Compiler Data Exec.

<table>
<thead>
<tr>
<th>Number</th>
<th>File Module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Type Name

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Lines and Comments

<table>
<thead>
<tr>
<th>Total Lines</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>209</td>
<td>78</td>
</tr>
</tbody>
</table>

### Compiler Data

<table>
<thead>
<tr>
<th>Lines</th>
<th>Comments</th>
<th>Direct</th>
<th>Instr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Data Structures

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Source Code

```
// Source code goes here
```

---
**"C_Outfile.dat" Summary Page**

C/C++ SOURCE LINES OF CODE COUNTING PROGRAM
(c) Copyright 1991, 1994 Northrop Grumman Corporation, CodeCount (TM)

Northrop Grumman Corporation retains ownership of this copy of software. It is licensed to you. Use, duplication, or sale of this product, except as described in the CodeCount License Agreement, is strictly prohibited. This License and your right to use the software automatically terminate if you fail to comply with any provisions of the License Agreement. Violators may be prosecuted.

This product is licensed to: Northrop Grumman Corporation (ESSD)

<table>
<thead>
<tr>
<th>Temporary Project Name</th>
<th>Total Blank</th>
<th>Comments</th>
<th>Compiler Data</th>
<th>Exec.</th>
<th>Number</th>
<th>File</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lines</td>
<td>Lines</td>
<td>Whole Lines</td>
<td>Lines</td>
<td>Lines</td>
<td></td>
</tr>
<tr>
<td>16030</td>
<td>2059</td>
<td>5111</td>
<td>1980</td>
<td>673</td>
<td>1097</td>
<td>7090</td>
</tr>
<tr>
<td>456</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>456</td>
<td>0</td>
</tr>
</tbody>
</table>

Number of files successfully accessed.............................. 12

Total Number of Deliverable Source Instructions (DSIs)........... 9316

Number of files with:
- Executable Instructions > 100 = 6
- Data Declarations > 100 = 6
- Percentage of Comments to DSIs < 60% = 6

Total occurrences of these reserved words:

<table>
<thead>
<tr>
<th>Compiler Directives</th>
<th>C/C++ Reserved Words</th>
<th>Lines</th>
<th>Lines</th>
<th>Lines</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define</td>
<td>FILE</td>
<td>222</td>
<td>49</td>
<td>goto</td>
<td>0</td>
</tr>
<tr>
<td>#undef</td>
<td>const</td>
<td>12</td>
<td>0</td>
<td>return</td>
<td>12</td>
</tr>
<tr>
<td>#dictionary</td>
<td>int</td>
<td>0</td>
<td>568</td>
<td>continue</td>
<td>0</td>
</tr>
<tr>
<td>#is</td>
<td>long</td>
<td>0</td>
<td>562</td>
<td>break</td>
<td>83</td>
</tr>
<tr>
<td>#ifdef</td>
<td>unsigned</td>
<td>0</td>
<td>0</td>
<td>if</td>
<td>625</td>
</tr>
<tr>
<td>#ifndef</td>
<td>short</td>
<td>0</td>
<td>0</td>
<td>else</td>
<td>242</td>
</tr>
<tr>
<td>#else</td>
<td>char</td>
<td>0</td>
<td>225</td>
<td>for</td>
<td>223</td>
</tr>
<tr>
<td>#elif</td>
<td>float</td>
<td>0</td>
<td>30</td>
<td>do</td>
<td>0</td>
</tr>
<tr>
<td>endif</td>
<td>double</td>
<td>0</td>
<td>0</td>
<td>while</td>
<td>208</td>
</tr>
<tr>
<td>#include</td>
<td>enum</td>
<td>30</td>
<td>19</td>
<td>switch</td>
<td>16</td>
</tr>
<tr>
<td>#line</td>
<td>class</td>
<td>0</td>
<td>0</td>
<td>case</td>
<td>86</td>
</tr>
<tr>
<td>#module</td>
<td>struct</td>
<td>0</td>
<td>6</td>
<td>default</td>
<td>5</td>
</tr>
<tr>
<td>#pragma</td>
<td>union</td>
<td>0</td>
<td>0</td>
<td>entry</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>void</td>
<td>0</td>
<td>99</td>
<td>sizeof</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>typedef</td>
<td>110</td>
<td>0</td>
<td>new</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>auto</td>
<td>0</td>
<td>0</td>
<td>delete</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>0</td>
<td>0</td>
<td>stdin</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>static</td>
<td>99</td>
<td>0</td>
<td>stdout</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>extern</td>
<td>0</td>
<td>0</td>
<td>stderr</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>globaldef</td>
<td>0</td>
<td>0</td>
<td>cin</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>globalref</td>
<td>0</td>
<td>0</td>
<td>cout</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>globalvalue</td>
<td>0</td>
<td>0</td>
<td>cerr</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>readonly</td>
<td>0</td>
<td>0</td>
<td>clog</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>noshare</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

This output produced on Fri Mar 7 17:41:58 1997

REVISION AU9  SOURCE PROGRAM -> C_LINES

NORTHROP GRUMMAN
- across multiple programming languages

**Portability issues achieved**
- VAX (VMS & Ultrix)
- SGI (UNIX)
- SUN (Sun OS & Solaris)
- HP (HP-UX)
- PC (Windows NT)

**Significant reduction in data gathering effort**

**Increased accuracy and consistency in sizing information**
LESSONS LEARNED

• Develop/Acquire software sizing tools that consistently applies the same LOC definition across programming languages

• Proliferate the same data gathering toolset across the enterprise

• Differentiate language-level counting rules (automated in tool) from project-level counting rules (policy & procedures)
  - decouple operation of LOC counting from project-level issues

• Subclassification of language constructs leads to:
  - difficulty in automation
  - decrease in understanding, confusion upon interpretation
  - misclassification
  - lower enterprise-wide utility

• Programming Languages will be revised
  - automated LOC counting tools must be enhancable
LESSONS LEARNED

• Limit complexity of software sizing tools
  - reduce parsing complexity via restricting analysis to source code that successfully compiles
  - decouple parsing / counting algorithms from coding standards & style requirements
  - decouple parsing / counting algorithms from tools that "preprocess" the source code input
  - decouple operation of the tool from project-level issues
    » what code to count
    » when to count the code
    » what is the efficiency or quality of the code
  - limit amount of LOC subclassifications to detect and report upon separately

• Software sizing tool should be:
  - easy to use
  - portable
  - support multiple end-user requirements
CONSULTED

- Robert E. Park, Software Engineering Institute
- Donald J. Reifer, Reifer Consultants Inc.