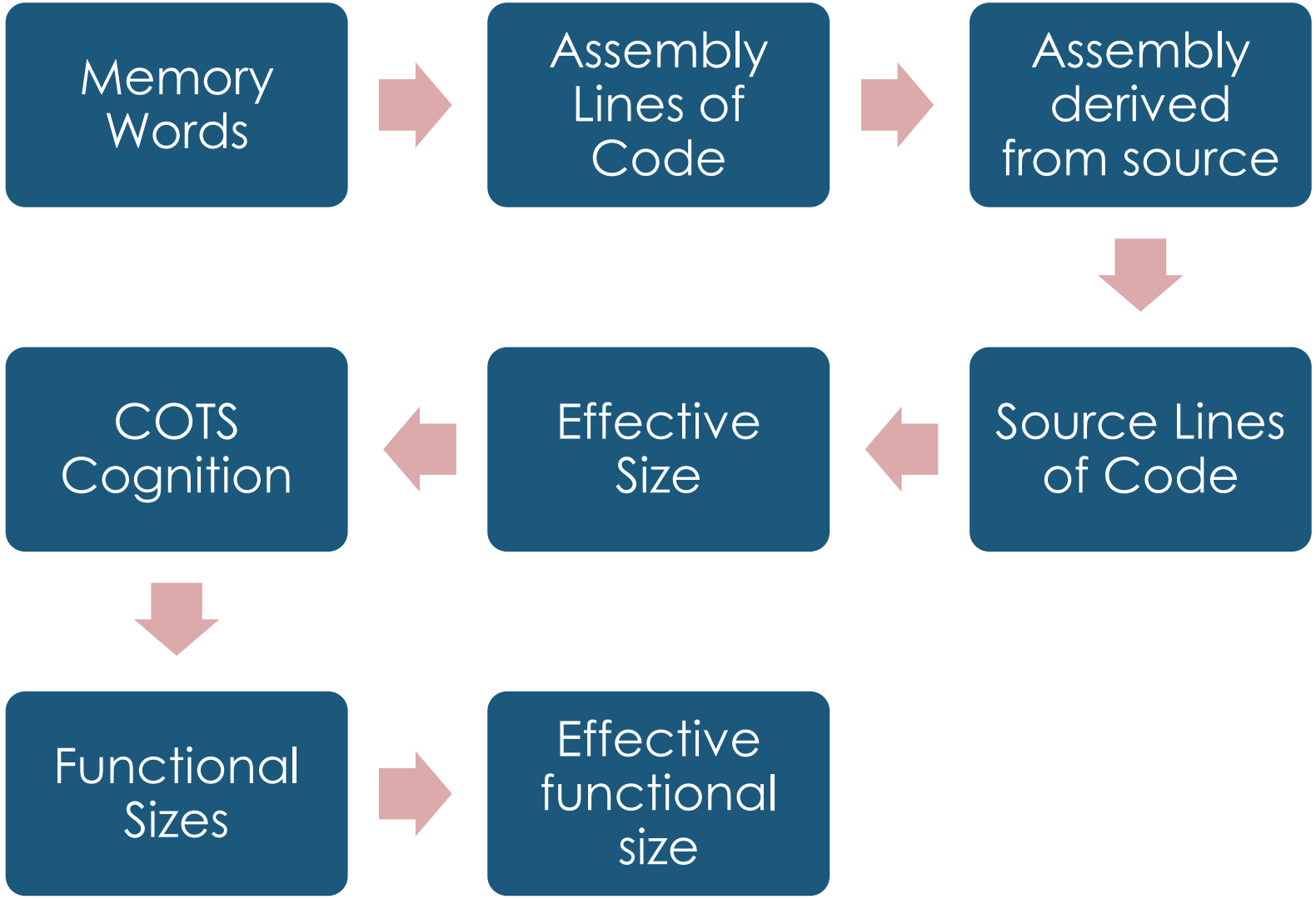
NESMA 2020

Software Sizing As An Essential Measure: Past, Present, and Future

GALORATH

Size Has Always Been Needed: Short History of Software Sizing



GALORATH

Historical Sizing Memory Size Illustration

- 64k bytes
- 20% reserve
- 52k bytes of code used
- Bytes per instruction (Machine dependent)
  - Least, Likely Most
  - Likely 6 bytes per instruction ~ 8700 SLOC

Source to Object Or Object To Source

- You could see the assembly code mapping in the original PDP 11 C Language

```
int get_int(int c);
int main(void) {
    int a = 1, b = 2;
    return getCode(a) + b;
}
```

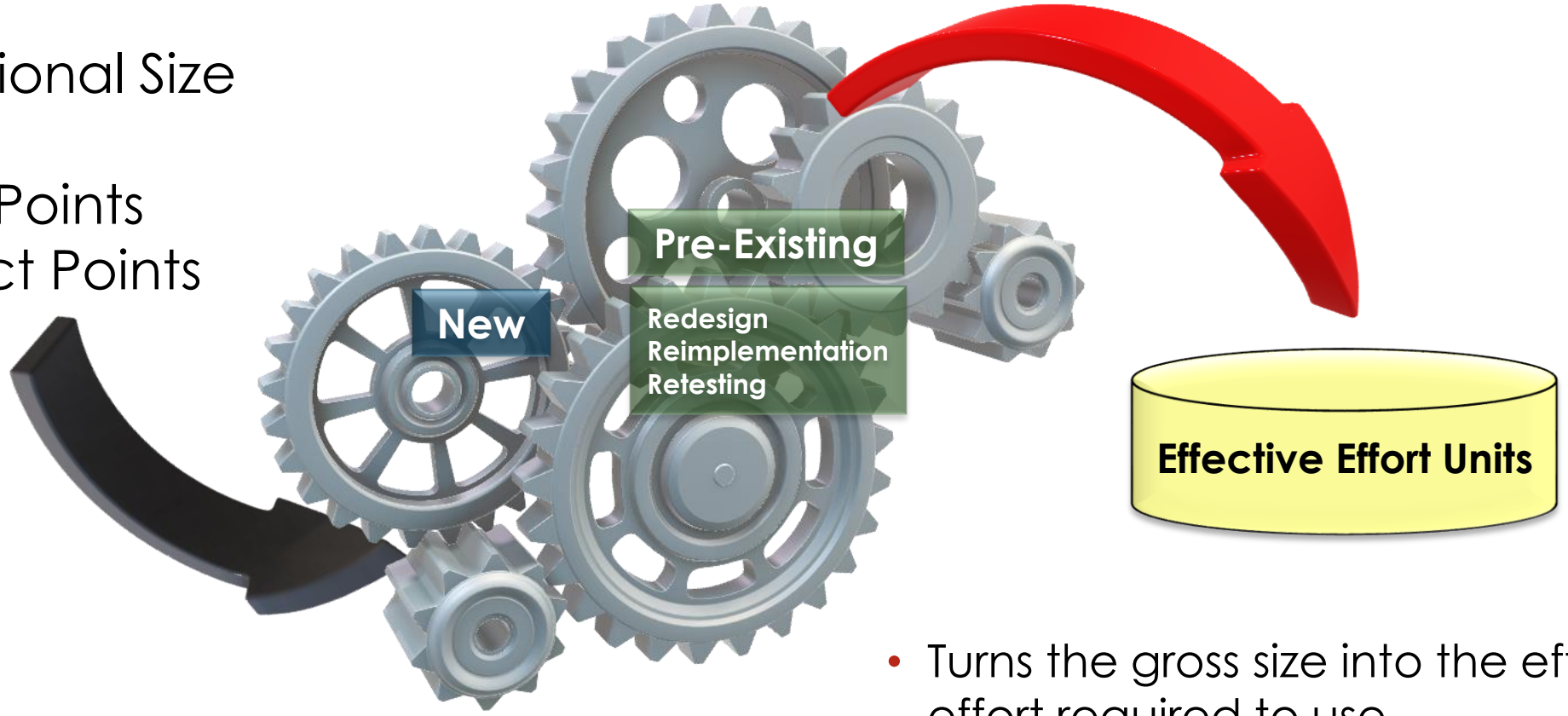
$ `gcc -c -g test.c`

$ `objdump -Sd ./test.o`

```
00000000 <main>:
int get_int(int c);
int main(void) { /* here, the prologue creates the frame for main */
   0:   8d 4c 24 04             lea    0x4(%esp),%ecx
   4:   83 e4 f0                and    $0xfffffff0,%esp
   7:   ff 71 fc                pushl  -0x4(%ecx)
   a:   55                      push   %ebp
   b:   89 e5                   mov    %esp,%ebp
   d:   51                      push   %ecx
   e:   83 ec 14                sub    $0x14,%esp
    int a = 1, b = 2; /* setting up space for locals */
  11:   c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%ebp)
  18:   c7 45 f8 02 00 00 00    movl   $0x2,-0x8(%ebp)
    return getCode(a) + b;
  1f:   8b 45 f4                mov    -0xc(%ebp),%eax
  22:   89 04 24                mov    %eax,(%esp)
  25:   e8 fc ff ff ff          call   26 <main+0x26>
  2a:   03 45 f8                add    -0x8(%ebp),%eax
} /* the epilogue runs, returning to the previous frame */
  2d:   83 c4 14                add    $0x14,%esp
  30:   59                      pop    %ecx
  31:   5d                      pop    %ebp
  32:   8d 61 fc                lea    -0x4(%ecx),%esp
  35:   c3                      ret
```

Software Size ⇨ Effective Effort Units

- Functional Size
- SLOC
- Story Points
- Object Points
- Etc…



- Turns the gross size into the effective effort required to use
- Amount of effort rather than total size

GALORATH

Effective Size To Normalize New and Pre-Existing

NEW

Life cycle activities will be performed at 100%

EXISTING

Effective size" based on user-provided rework percentages.

Results in a "reuse benefit" that reduces the overall effort and schedule estimates

REWORK MULTIPLIER		0	0	0	
Step 1: Set Reverse Engineering & Redesign Factors					
Redesign Breakdown					
	Formula	0.22*A+0.78*B+0.5*C+0.3*(1-(0.22*A+0.78*B))*(3*D+E)/4			
	Result Redesign Percentage	0.00%	0.00%	0.00%	
Weight	**Redesign Component**	**Least**	**Likely**	**Most**	**Percentage of the existing software that...**
0.22	Architectural Design Change	0%	0%	0%	GENERALLY 0 for OPEN SOURCE
0.78	Detailed Design Change	0%	0%	0%	GENERALLY 0 for OPEN SOURCE
0.5	Reverse Engineering Required	0%	0%	0%	COTS COGNITION
0.225	Redocumentation Required	0%	0%	0%	... requires redocumentation
0.075	Revalidation Required	0%	0%	0%	... requires revalidation with the new design
Step 2: Set Reimplementation Factors					
Reimplementation Breakdown					
	Formula	.37* A + .11*B + .52*C			
	Result Reimplementation Percentage	0.00%	0.00%	0.00%	
Weight	**Inputs**	**Least**	**Likely**	**Most**	**Percentage of the existing software that...**
0.37	Recoding Required	0%	0%	0%	GENERALLY 0 for OPEN SOURCE
0.11	Code Review Required	0%	0%	0%	GENERALLY 0 for OPEN SOURCE
0.52	Unit Testing Required	0%	0%	0%	GENERALLY 0 for OPEN SOURCE
Step 3: Set Retest Factors					
Retest Breakdown					
	Formula	.10*A + .04*B + .13*C + .25*D + .36*E + .12*F			
	Result Retest Percentage	0.00%	0.00%	0.00%	
Weight	**Inputs**	**Least**	**Likely**	**Most**	**Percentage of the existing software that...**
0.1	Test Plans Required	0%	0%	0%	... requires test plans to be rewritten
0.04	Test Procedures Required	0%	0%	0%	... requires test procedures to be identified and written
0.13	Test Reports Required	0%	0%	0%	... requires documented test reports
0.25	Test Drivers Required	0%	0%	0%	... requires test drivers and simulators to be rewritten
0.36	Integration Testing	0%	0%	0%	... requires integration testing
0.12	Formal Testing	0%	0%	0%	... requires formal demonstration testing
Step 4: Copy Data Into SEER-SEM					

Reports

Quick Estimate | **Software Metrics**

Metric	Functions	Effort Units
Effective Size	137	2,055
Total Size	201	2,940
New Size	76	1,126
Reuse Benefit	31%	30%
Cost Per Size Unit	603	40
Effective Defect Density	0.06	4.25
Total Defect Density	0.04	2.97

GALORATH

Most Popular Languages Have Standard Line Terminator: Matches the Detailed Ruled

```
procedure Example;

(*This is a comment; not a line of code*)
Const sds = 321;
Type Z = array[1..44] of CHAR;
Var A,B        : Integer;

begin
   if (A = 22) then
       B := 4;
   A :=                (* set it right!! *)
       (B + sds)/2;

end;
end; (*example*)
```

Partial	1
Blank	2
Comment	3
1 Declaration	4
2 Declaration	5
3 Declaration	6
Blank	7
Partial	8
Partial	9
4 Source	10
Partial	11
5 Source	12
Blank	13
6 Source	14
7 Declaration	15

Logical SLOC Includes

- Executable source lines
- Control (DO WHILE, REPEAT UNTIL, ...)
- Mathematical (i:=a**b/c;)
- Conditional (IF, THEN, ELSE)
- Input/output & formatting
- Data declarations (including data typing & equivalence)
- Deliverable script

Logical SLOC Does *NOT* Include

- Comments & blanks (part of documentation)
- BEGIN statements from BEGIN-END pairs (pair counts as one)
- Continuation separate physical lines for convenience
- Machine/library generated statements (lines instantiated)
- Non-final test code or debug statements (part of testing)

15 Carriage returns but only 7 logical lines

Simple counting… difficult estimating unless you have size database

GALORATH

Key Components Of A Software Project That Uses Commercial Off the Shelf Software)COTS)

Developmental Software:

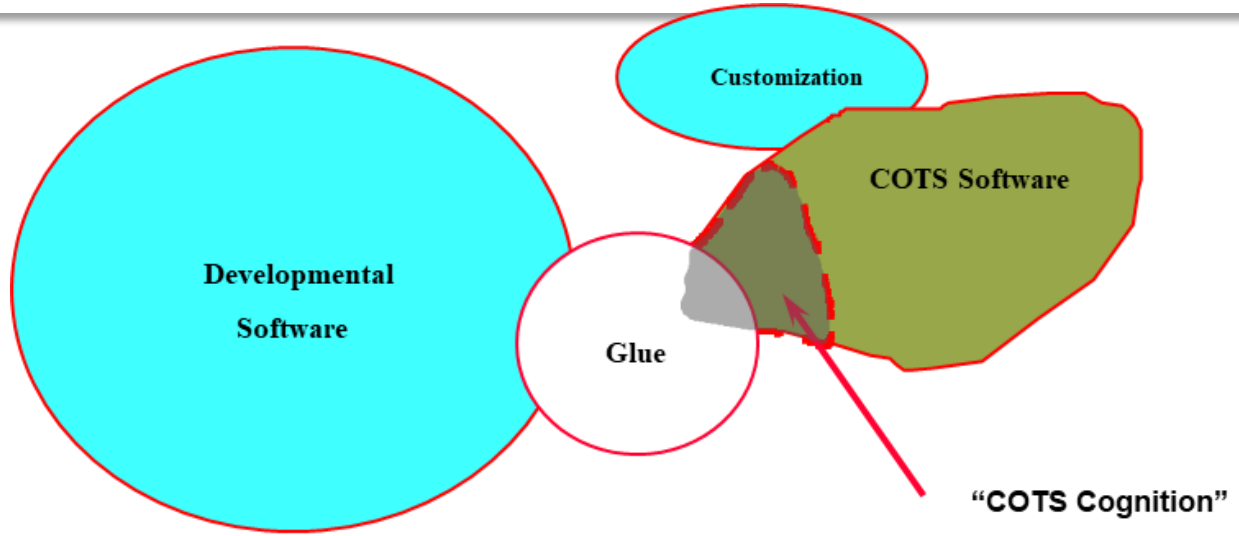
- Functionality developed specifically for the project at hand
- May include customization of COTS

"Glue" Code:

- Code written to bind COTS to developmental software
- Development effort must be captured

Cots Cognition

- Features
- Quick Size
- Objects



COTS Software:

- Purchased functionality
- Direct Cost component of COTS integration

COTS Cognition:

- Required functionality within the COTS software that must be understood
- Effort component of COTS integration

GALORATH

COTS Cognition

Features

- Unique Functions (Inputs, Outputs, Queries, Etc.)
- Tables referenced
- Tables configured

Quick Size

- Application type
- Percentage required

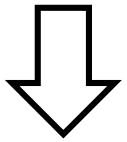
Object Sizing

- Input services, output services
- Inquiry services
- External classes
- Internal classes

Measurement and Quality of Open Source

- Coverity free size & quality service for open source
- Coverity Scan Open Source Report Shows Commercial Code Is More Compliant to Security Standards than Open Source Code

741 projects
2.5M LOC

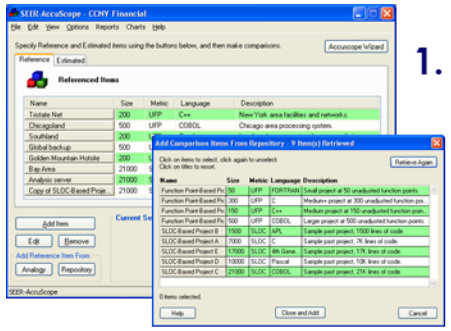


Coverity Scan

44,641 defects are fixed

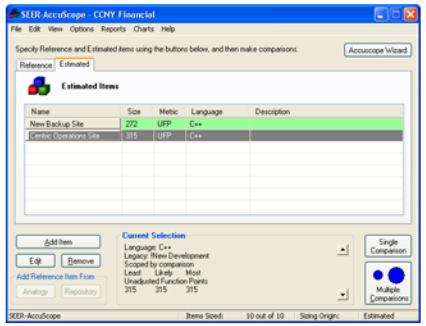
(Only 10.2% of identified defects are false positives in 2013)

Comparison Sizing Can Be Very Quick and Effective

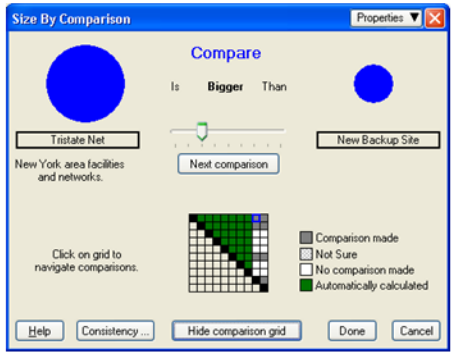


1. Choose reference items for comparison

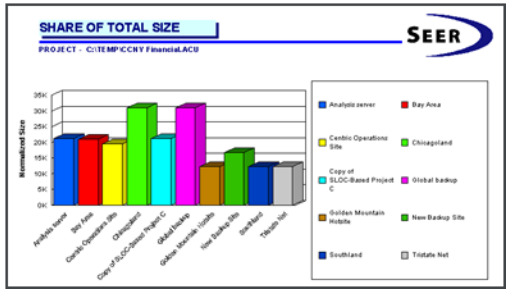
2. Identify Items to size



3. Perform various expert judgment comparisons



4. View / Use results



GALORATH

Conclusions

- There are MANY viable ways to count or estimate size
- Standards are key (ISO or not)
- Functional size is helpful, if standard are used, whether counted or estimated
- Don’t dismiss lines of code, counted with standards when appropriate
- Keep seeking better and better approaches

GALORATH

Contact

- Dan Galorath
- 1 310 414 3222 x614

- galorath@galorath.com

Scoping COTS Cognition Using Features and Objects

Sizing with Features is closely related to the traditional function point method

Measures have been translated so that they are closely related to features found in COTS components

Unique Functions (Inputs, Outputs, Queries, etc.)

Data Tables Referenced

Data Tables Configured

Use Objects if COTS components are implemented as objects

Object sizing metric is completely compatible with IFPUG-standard function points and object counting guidelines

Input Services

Output Services

Inquiry Services

External Classes

Internal Classes

Application Type

Commercial application or application type being integrated

If specific application is not listed, choose one that is similar in size/functionality to the application being integrated

Percentage Required

Portion of COTS component functionality that the integrating developers are required to learn

Reflects effort to acquire a working knowledge of this functional portion

GALORATH