U.S. AIR FORCE

What Now? – Auto-generated Source Code

Dr. Randall Jensen
Daniel Keth
Software Technology Support Center
Hill AFB, UT

History of autocode generation

- Fortran I completed in 1957 (John Backus)
  - Much more efficient than Assembler
  - Close to problem space
  - Required less programming skill
  - Portable code
- Refinement of languages (1957+)
- Prolog (1972) first logic language
- Smalltalk (1975) first object oriented language
- Visual Basic (1991)
- Universal Modeling Language – UML (1994)
  - Rhapsody (2004)

Effective source size

- Effective source size is a measure of the work required to produce a software product
- Software development cost is NOT directly related to generated code
  - Compiled object code
  - Auto-generated Source (C++) code
- Effective size is THE major cost and schedule driver in modern software cost estimating tools

Effective size and Work

- Difficult to relate PHYSICAL things to ABSTRACT work
- Work = Effort to design, implement and test

- General software development equation

$$E_d = C_s C_{env} S_{eff}^{1.2}$$ Person months

where

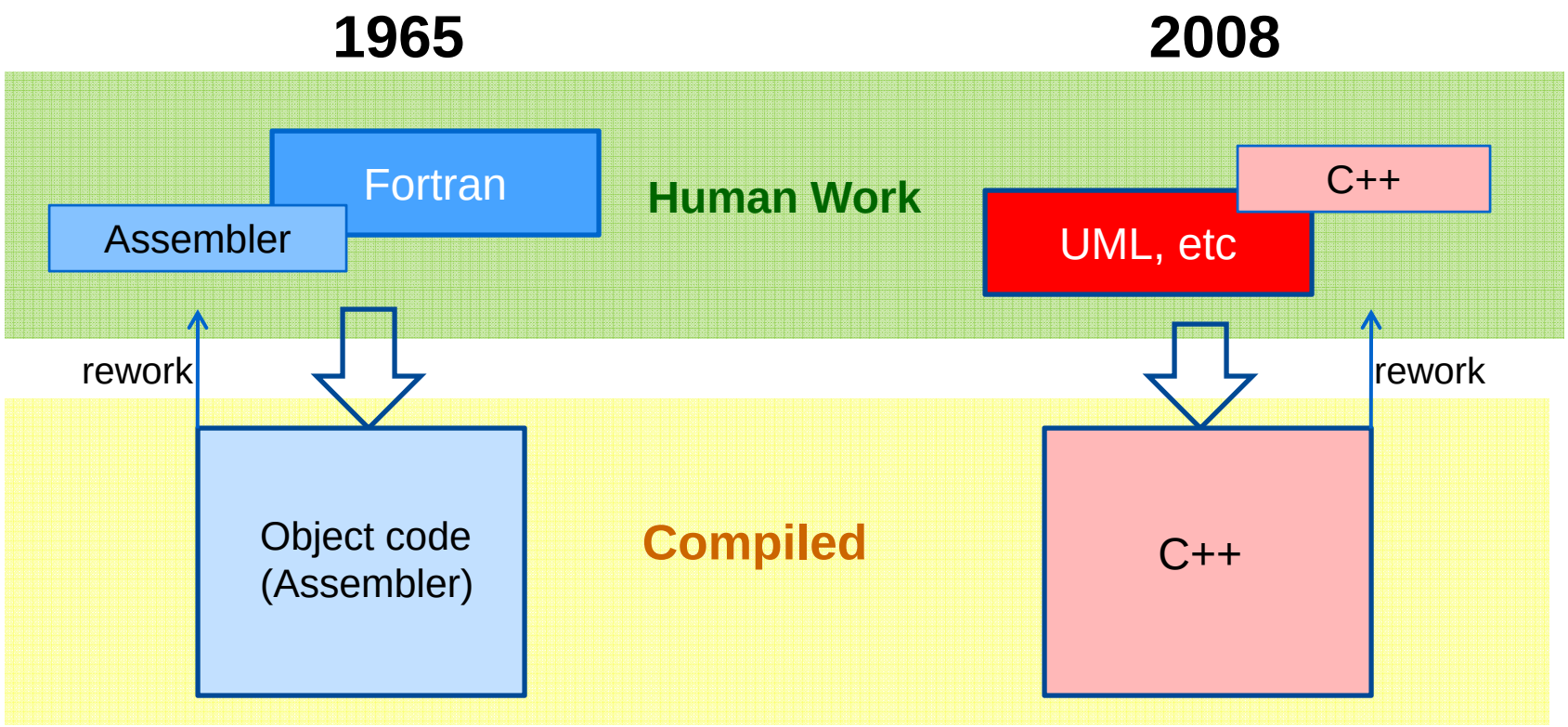
$C_s =$ Scaling constant

$C_{env} =$ Environment constant

$S_{eff} =$ Effective software source size

WORK

Source code comparison



Count should reflect "human" work performed.
Human work can be derived from ESLOC.
Compiled code is the product, NOT the work measure.

Sample auto-gen source code

Dave generated SLOC

```
Private sub circle area (x)
‘ computes area of circle given
‘ radius x
   Circlearea = PI*x^2
   Exit sub
End sub
```

HAL generated SLOC

```
Private sub rxy127(x)
   rxy127=ct2780*x^2
   Exit sub
End sub
```

Auto-gen code example I

- SIAP characteristics:
  - 800,000 estimated lines of C++ code
  - Generated from UML/State chart model
  - Compiler is 90 percent efficient (C++ model)
  - Required modifications (refinements) = 25 percent
- What is effective size?
- Programmer: Dave
- Compiler: HAL 2008

Auto-gen code example II

Dave generates

- 2000 UML/StateChart statements
- 80,000 C++ statements
- 180,000 C++ modified statements
- 540,000 C++ reused statements

HAL generates

- 720,000 C++ statements

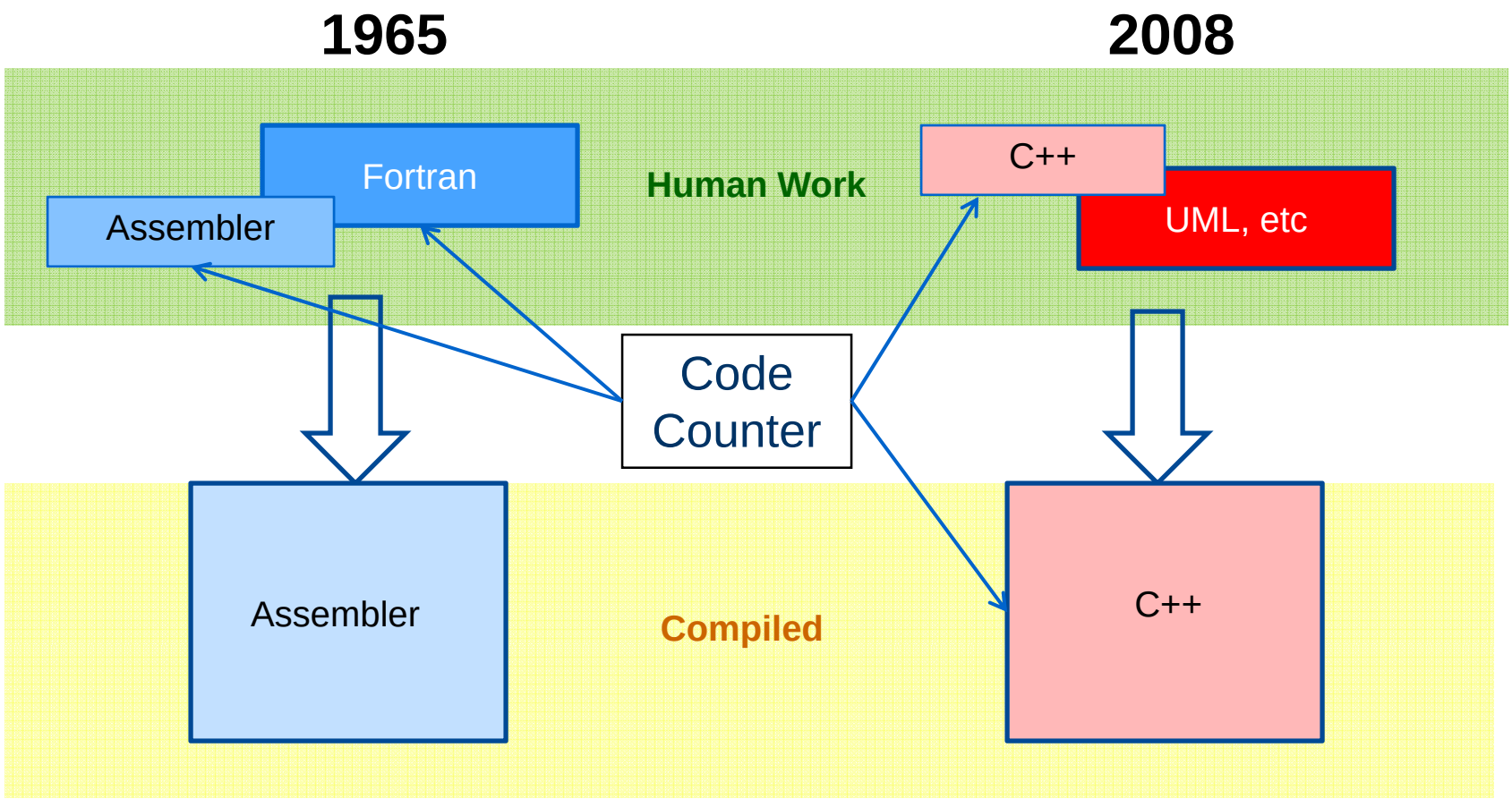
Auto-gen code example III

- What is effective size from auto-gen'd code?
  - $S_{eff} = S_{new} + 0.5S_{\text{mod}} + 0.05S_{reused}$
  - Obviously not!
- **Size depends on who creates code**
- Dave-generated code
  - $S_{eff} = S_{new} + 0.5S_{\text{mod}}$
- HAL-generated code
  - $S_{eff} = S_{\text{mod}} + \alpha S_{reused}$
  - Unknown code, new developer
  - Lower productivity

A modified auto-gen problem

- What happens when the UML/StateChart model is modified?
- A new 720,000 SLOC compilation is produced
  - How many UML statements were changed?
  - Is the entire 720,000 C++ modified?
- Modifications/corrections must be re-developed for the new compilation
- HAL-generated code
  - $S_{eff} = S_{\text{mod}} + \alpha S_{reused}$
  - Unknown code, new developer
  - Lower productivity

Code counters in the process



Count should reflect "human" work performed

Code counter issues

- Counter counts total, not effective code

$$S_{total} = S_{new} + S_{\text{mod}} + S_{reused}$$

$$S_{effective} = S_{new} + 0.5S_{\text{mod}} + 0.05S_{reused}$$

- Count equals effective code only if total and new code are identical AND it is hand coded
- Auto-gen code is not relevant to development effort
  - User-modified auto-gen code IS relevant
  - Reused auto-gen code IS relevant in presence of user modifications (retest)
  - Counters, by themselves, do not generally count modified/reused code correctly

Counter issue example

Project Data	
Size, new, SLOC	65,251
Size, modified, SLOC	1,097,565
Size, reused, SLOC	481,264
Development effort, pm	170
Development schedule, mo	5
Productivity, SLOC/pm	3,753

Summary

- Technology to size or scale UML and state charts is not available yet.
- No historic data to support estimates of the impact of modifying auto-generated code.
- Code counters are mis-applied by counting generated code as a cost estimating (effective size) measure.
- Do we count the object code generated by Fortran and Ada compilers as a measure of productivity?