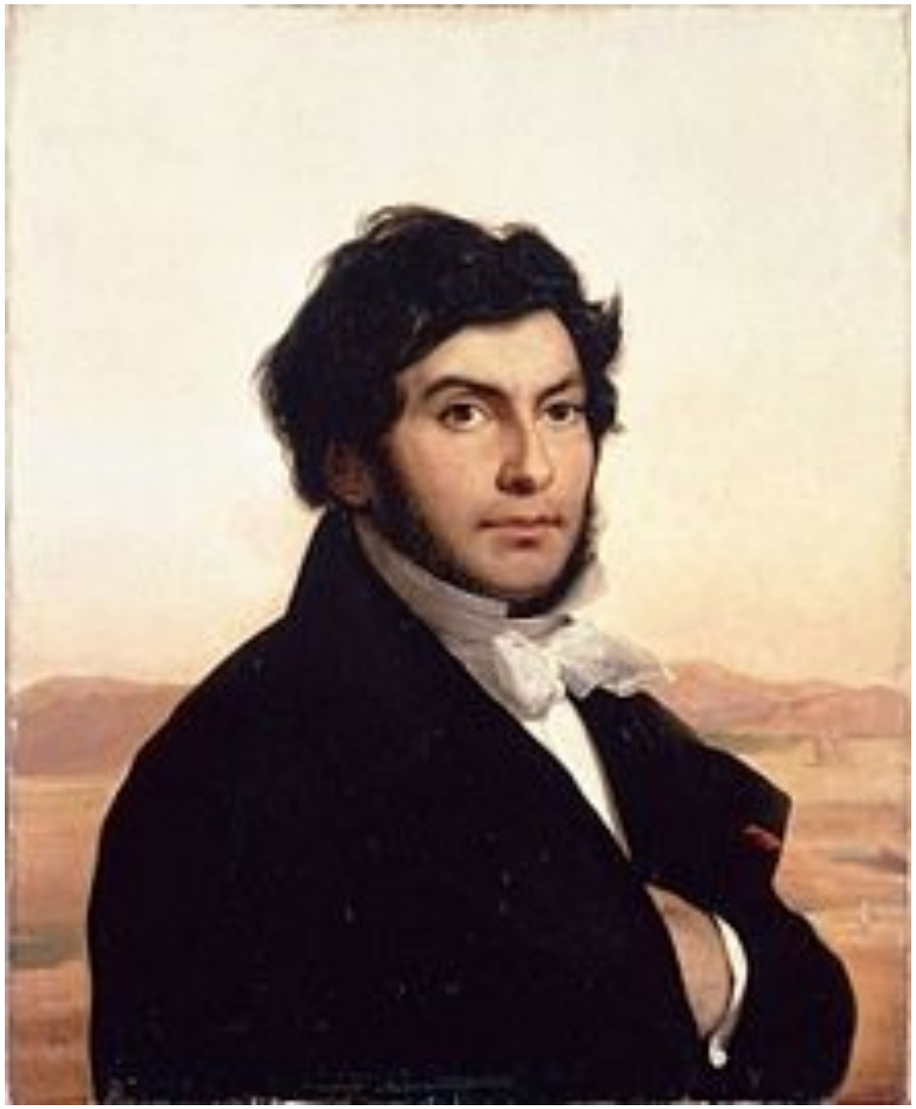
Champollion meets Apollyon: What Makes Empirical Software Engineering so Hard?

Richard A. O'Keefe,
18 March 2016

Champollion

- Jean-François Champollion
- Ancient Egyptian had three scripts: hieroglyphic, hieratic, and demotic. Coptic used a variant of Greek; readable.
- Champollion figured out how to read hieroglyphic.



The Rosetta Stone

- A decree of the priests of Memphis
- Written in Greek, in Egyptian demotic, and in hieroglyphs.

Simplistic Significance

- Having the *same* text written three ways
- meant that knowledge of one language and script (Greek)
- and a closely related language (Coptic)
- could be used to bootstrap Ancient Egyptian and its scripts

RosettaCode

- Rosetta Code is a programming chrestomathy site. The idea is to present solutions to the same task in as many different languages as possible, to demonstrate how languages are similar and different, and to aid a person with a grounding in one approach to a problem in learning another. Rosetta Code currently has 782 tasks, 193 draft tasks, and is aware of 609 languages, though we do not (and cannot) have solutions to every task in every language.

Empirical Software Engineering

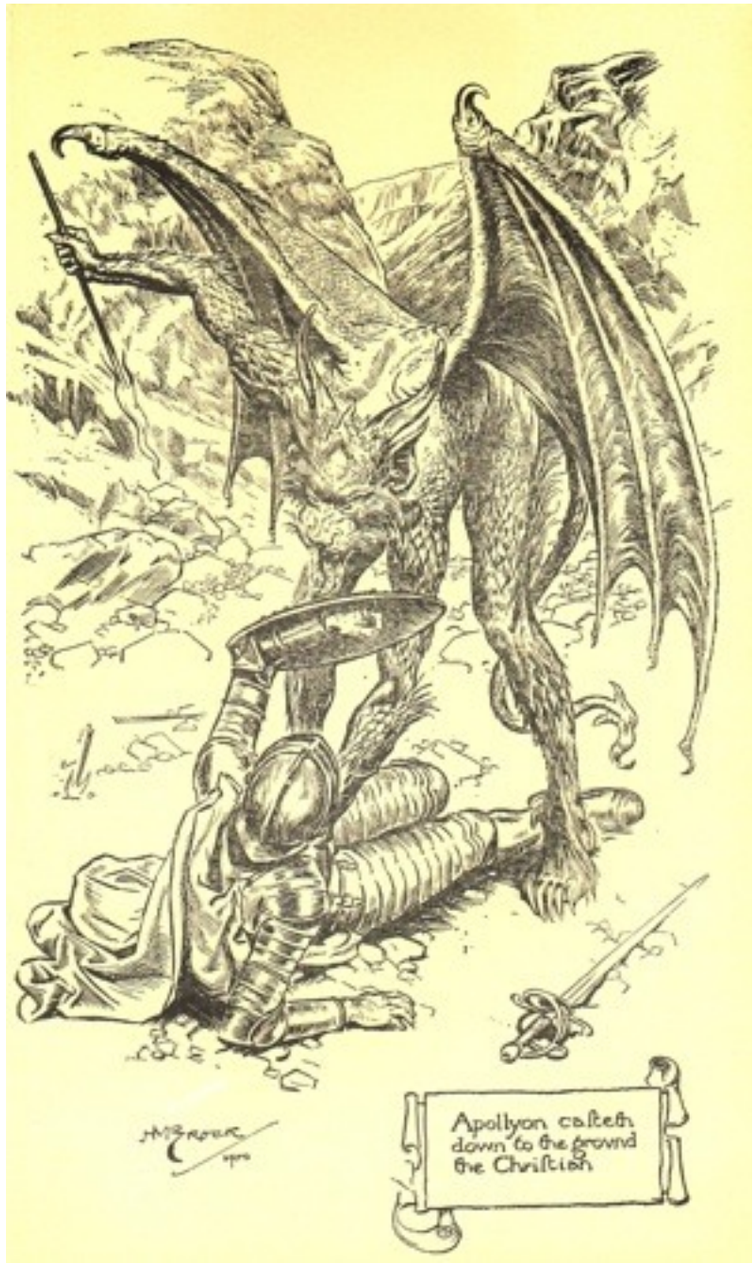
- Programming languages are different.
- People have opinions about which is better.
- (Not that there's only one kind of "better")
- Wouldn't it be nice to base our opinions on *empirical measurements?*

Lutz Prechelt

- "An empirical comparison of seven programming languages", IEEE Computer, Volume 33, issue 10, October 2000.
- (C, C++, Java) *vs* (Perl, Python, Tcl, Rexx)
- Scripting languages "often turn out better than Java and not much worse than C(++)"

Apollyon

- "The Destroyer", the king of the army of locusts in Revelation 9:11.
- The foe of Pilgrim in the Valley of Humiliation in Bunyan's *Pilgrim's Progress*



What is the destroyer?

- Prechelt: *In general, the differences between languages tend to be smaller than the typical differences due to different programmers within the same language.*
- Time: bad/good from 1.5 for Tcl to 27 for C++, Memory: 1.2 for Python to 4.9 for C++, Length: 1.3 for C to 3.7 for Rexx

Prechelt: limitations

- Prechelt had *one* problem.
- Prechelt had only 80 programs.

Nanz and Furia

- Experiments take much time and money
- Wouldn't it be nice if someone else had already done the hard work?
- Oh look, someone *has!*
- Let's analyse RosettaCode.

Where do I come in?

- I've been working on a Smalltalk compiler and library for several years.
- Last year I implemented 80% of the RosettaCode problems in Smalltalk.
- Nanz and Furia's analysis of RosettaCode looked great, *but*

Remember Apollyon?

- Nanz and Furia wanted to compare
  - Program length
  - Run time
  - Memory
  - Reliability
- for C, Go, C#, Java, F#, Haskell, Python, Ruby

Prechelt met the Destroyer

- Variability between programmers exceeds variability between languages (one problem).
- Nanz & Furia cite Prechelt, but omit discussion of this point.
- Is it true of Rosetta Code?

Language Effect

(how much shorter than C)

- 1.00 ×/÷ 1.12 C
- 1.07 ×/÷ 1.12 Go
- 1.23 ×/÷ 1.12 Java
- 1.37 ×/÷ 1.12 C# (remember this)
- 1.49 ×/÷ 3.15 AWK
- 1.50 ×/÷ 1.12 JavaScript
- 1.52 ×/÷ 1.11 Ruby
- 1.54 ×/÷ 1.11 Python
- 1.98 ×/÷ 1.11 Haskell

Task Effect

(how much shorter than median task)

- Min 0.26 (4 times as long)
- 1st quartile 0.67 (1.5 times as long)
- Median 1.00 (by definition)
- 3rd quartile 1.49 (2/3 as long)
- Maximum 10.79 (9% as long)

Model

- Fit log(SLOC) = f(Language) + g(Task)
- Report as SLOC = f'(Language) × g'(Task)
- Take *cum grano salis*

Programmer Effect

Longest/Shortest ratio

Language	Q1	Median	Q3	Max	N
C++	1.24	1.67	2.33	15.55	87
C	1.32	1.75	2.47	7.67	144
C#	1.24	1.46	2.11	10.38	55
Java	1.27	1.56	2.33	6.52	90
JS	1.15	1.63	2.24	6.00	47
Haskell	1.22	1.54	2.19	14.75	82
Ruby	1.27	1.60	2.07	4.79	85
AWK	1.28	1.38	1.73	1.88	12
Python	1.20	1.57	3.32	30.16	151
Go	1.20	1.62	1.62	5.35	116
SML	1.15	1.22	1.22	1.38	3

How obtained

- Consider (language, task) pairs having multiple solutions
- Determine length of each solution
- Record longest/shortest
- Summarise by language

Apollyon victorious

- Programmer effects are greater than language effects.
- For better analysis, we need to know who wrote what.
- Rosetta Code records that but it's hard to get; Nanz and Furia took a snapshot but threw that away.

Problems with SLOC

- How do you measure the length of a program?
- Nanz and Furia used “cloc” because it handles dozens of languages.
- It counts every line containing a token of the language (*i.e.*, replace comments by spaces then discard blank lines)

Cloc reports 4 sizes

if (c)
  {
  x++;
  }
	if (c) {
	  x++;
	}

if (c) x++;	if (c) x++;

Layout effect

Language	“Tight”	Original	“Loose”
C	1.00	1.06	1.24
Java	1.00	1.03	1.27
C#	1.00	1.41	1.44

Data: unclean, unclean!

- In the snapshot, test data, sample output, transcripts &c are mislabelled as source.
- Languages where source & module name must match (Erlang, Java, &c) don't so can't be compiled without manual correction
- Code may not compile anyway (unspecified or wrong dialect, or just plain wrong).

Arbitrary-precision integers example

- Compute 5**(4***(3**(2**1)))
- Since 4**(3**(2**1)) = 262,144, this reduces to Compute 5**262144
- How is testing *one library function* in a particular implementation a test of a *programming language*?

NOT the same problem

- Example: Matrix Arithmetic says to compute determinant and permanent but not *how*. Direct algorithm O(n.n!), Ryser algorithm O(n.2^n) but naively O(n^2.2^n)
- Example: most "currying" solutions aren't
- Example: flatten a list, some versions discard duplicates and some don't.