“Coding with meaningless symbols"

Version 1.3: 08-June-2022

Les Hatton
Emeritus Professor of Forensic Software Engineering,
Kingston University, London

Overview

- The future of coding in safety-critical systems
- Coding and the behaviour of systems
  - Setting the scene: the user's point of view
  - What have we learned about coding?
- Coding and meaningless symbols
  - Emergent and apparently inevitable properties of all discrete systems
- The future of coding in safety-critical systems (reprise)

The future of coding in safety-critical systems

- Notice how “coding” has replaced “programming”
  - Coding is easy – you don’t have to think,
  - Anybody can do it – a week’s course and off you go,
  - Coding bears the same relationship to programming as a garden shed does to the Forth railway bridge.
- The future of coding
  - Presumably more of the same.
- The future of safety-critical systems
  - The absorption of safety-critical development into the morass of “normal” development.

The future of coding: More of the same?

- We have the standardised and ever-growing behemoths, Ada, C, C++, Fortran …
- A frisson of “new” ones accompanied by unsubstantiated claims for their efficacy, Perl, Python, Ruby, Rust, C# … the list is endless. Relative usage is determined by who shouts the loudest.
- Nobody ever measures anything in a falsifiable form so suitability for safety-criticality ? Who knows?

The future of coding:
Setting the scene

- Boeing 737-Max. How Boeing trashed 60 years of engineering reputation
- Cars: Keyless or Clueless? That was a stupid idea, let’s do it again.
- How to deploy an online system, 2022 style. We screw this up the same way every time. What fun!

The future of coding:
What have we learned about coding?

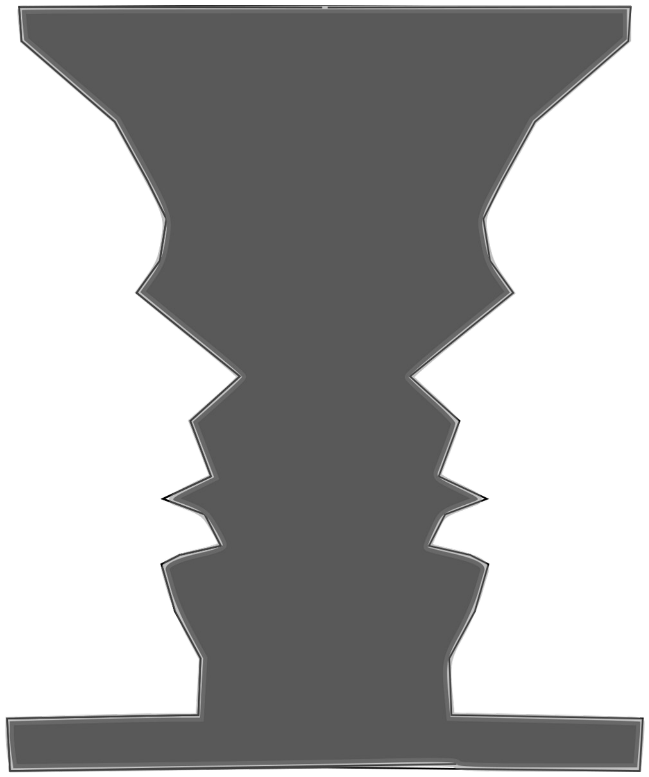
- Boeing 737-Max …
- Cars: Keyless or Clueless? …
- How to deploy an online system, 2022 style. …

All involve software but coding had nothing to do with any of these debacles. For a really comprehensive screw-up, you need management.

The future of safety-critical systems:

- Blurring with “conventional” systems
- Budget, budget, budget.
- Management ignorance: “a coder can code anything”. Discuss.
- Ambiguity of appropriate technologies

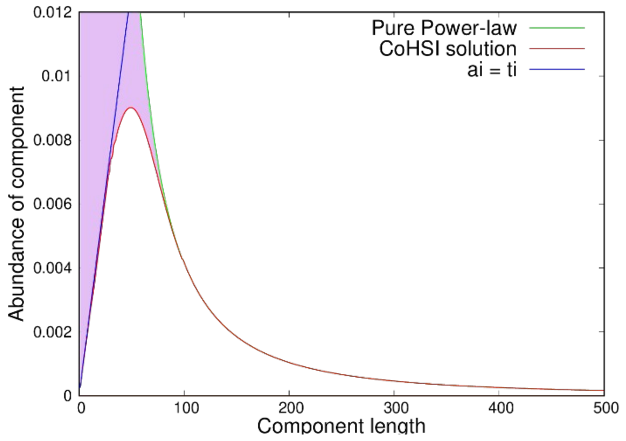
Coding and meaningless symbols: Multiple perspectives



Coding and meaningless symbols:
Emergent properties of all symbol systems

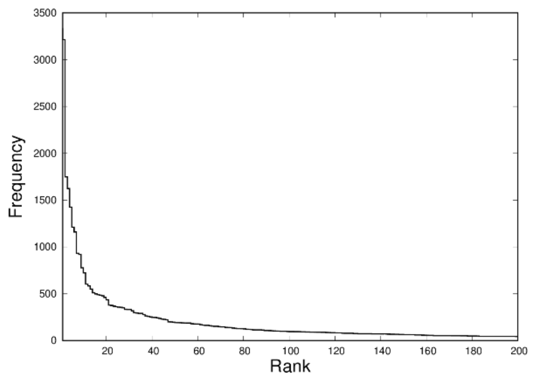
No implied ordering

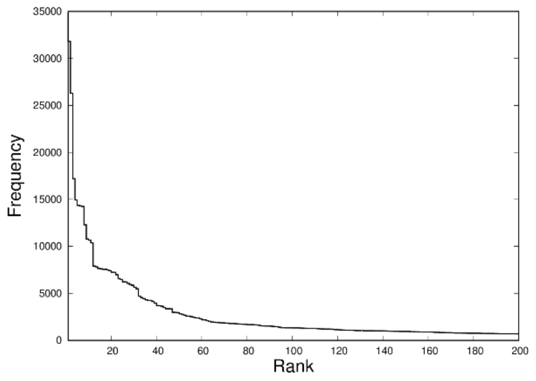
Implied ordering

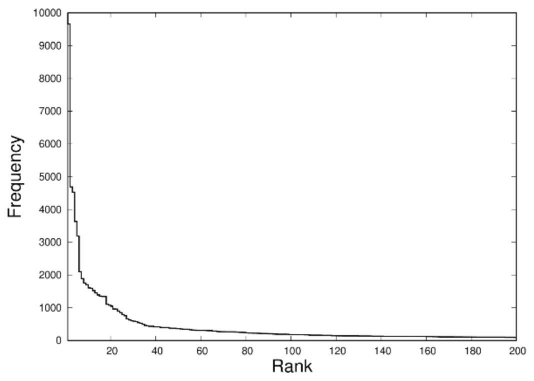


Coding and meaningless symbols:
… when beads **in boxes** represent words

- Zipf’s law and books; really spooky







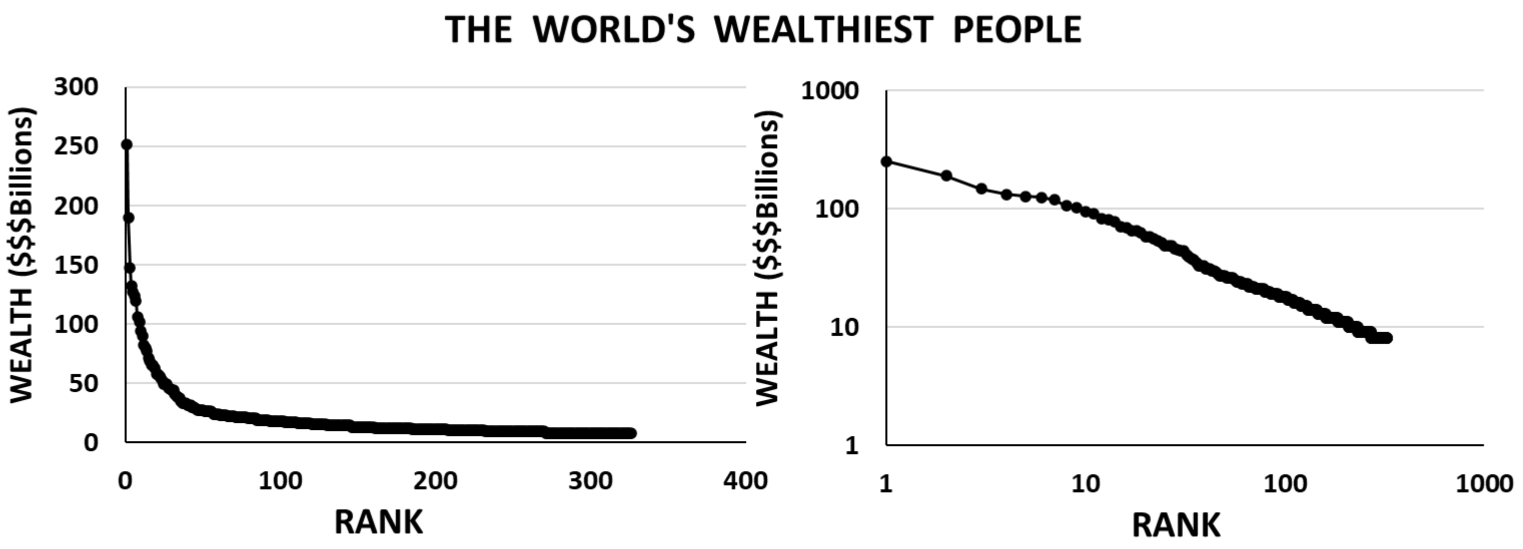
Three Men in a Boat (English)

King James Bible (Swedish)

European Constitution (Klingon)

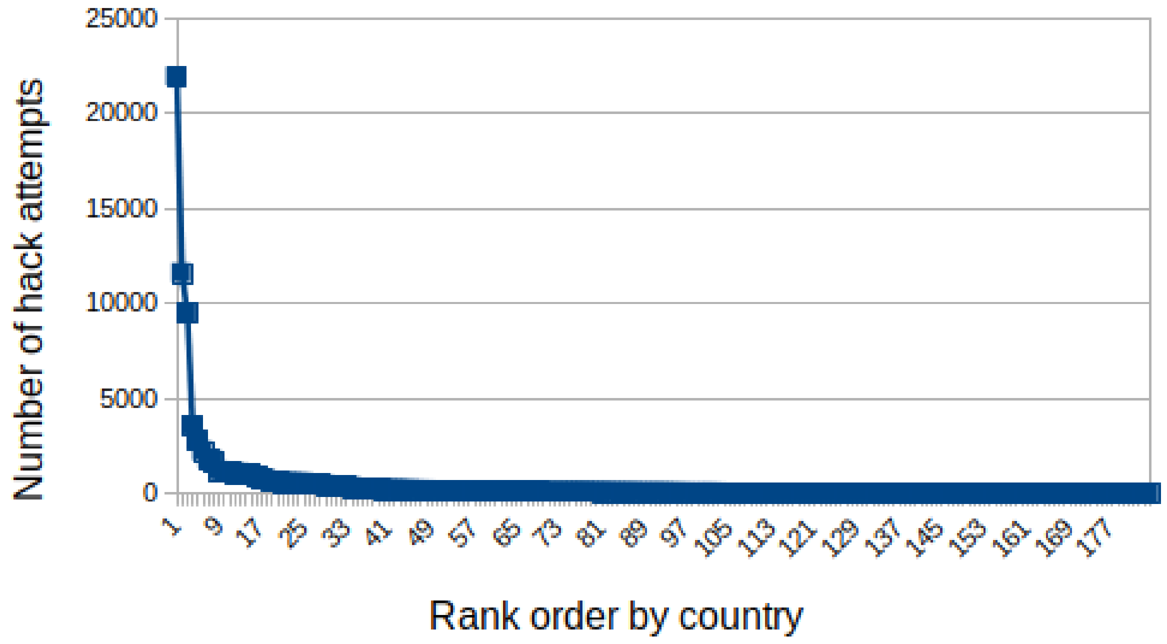
Coding and meaningless symbols:
… when beads **in boxes** represent wealth

- Pareto and wealth; also spooky



Coding and meaningless symbols:
… when beads **in boxes** represent hacking attempts

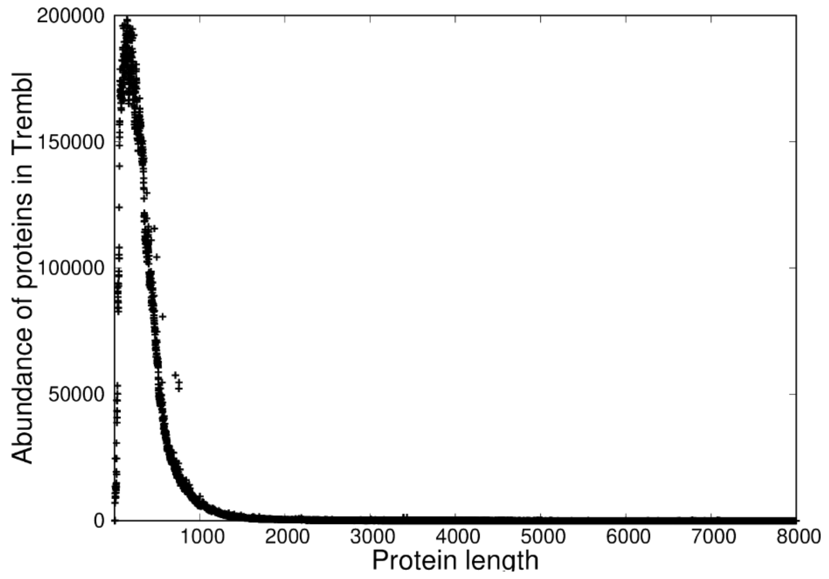
- Hacking attempts on tripwire server; also spooky



Coding and meaningless symbols:

… when beads **on strings** represent amino acids

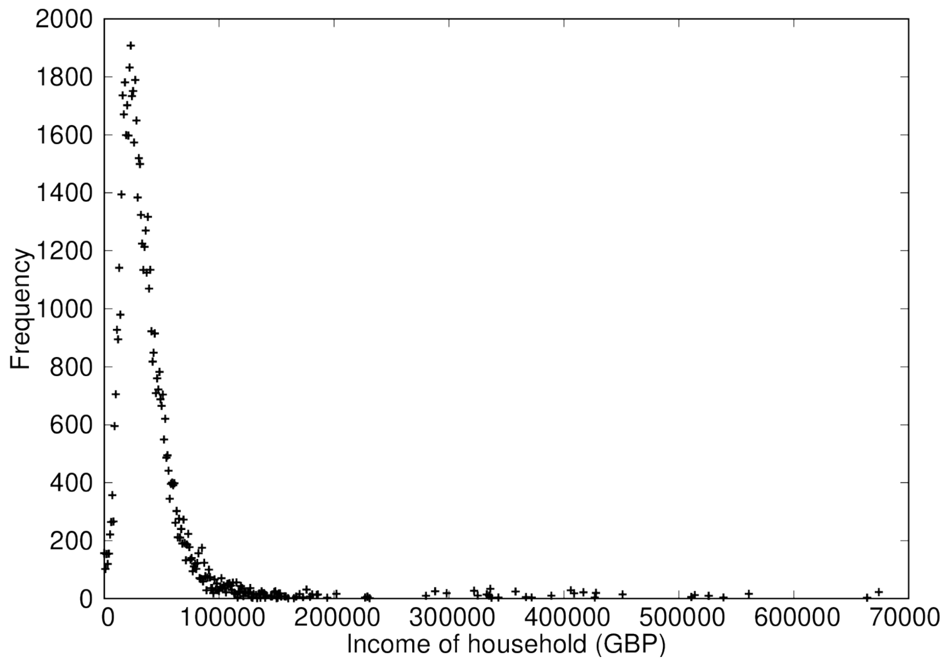
- Proteins; really spooky



~100 million proteins

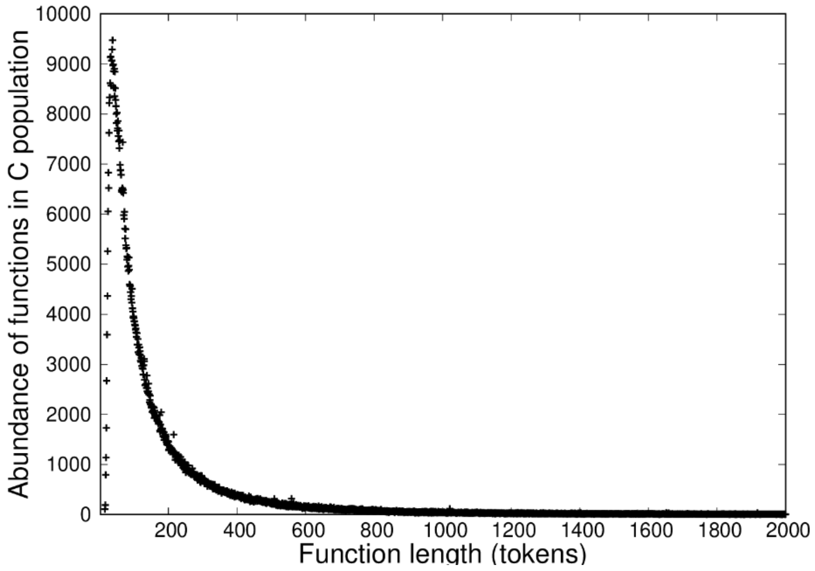
Coding and meaningless symbols: … when beads **on strings** represent currency amounts

- UK household wealth; also spooky



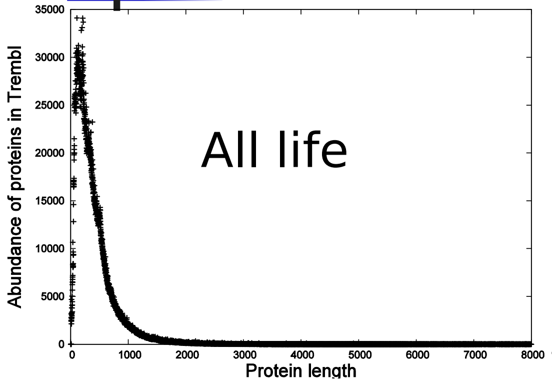
Coding and meaningless symbols: … when beads **on strings** represent language tokens

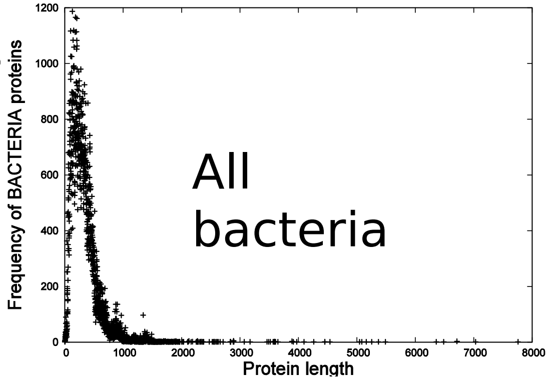
- Programs; also spooky

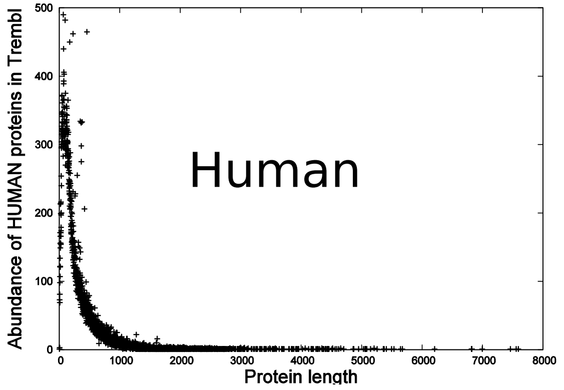


~100 million lines of code

Scale invariance - proteins

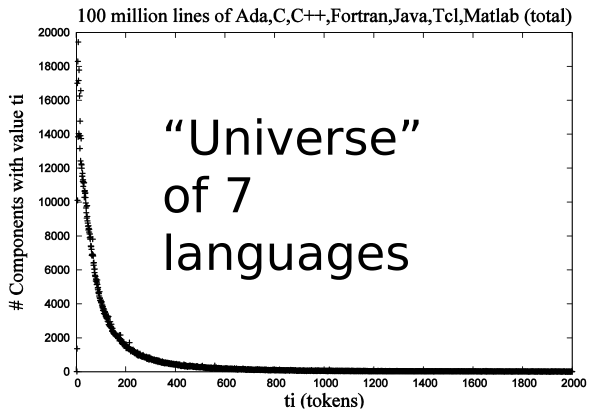


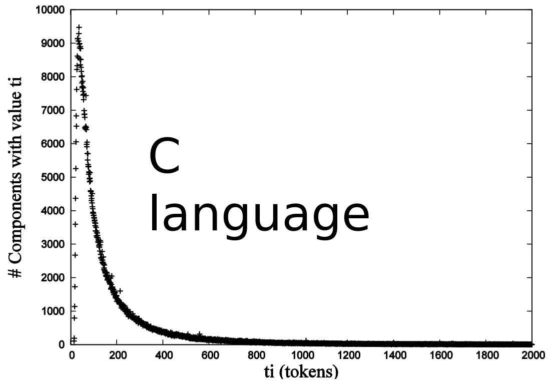


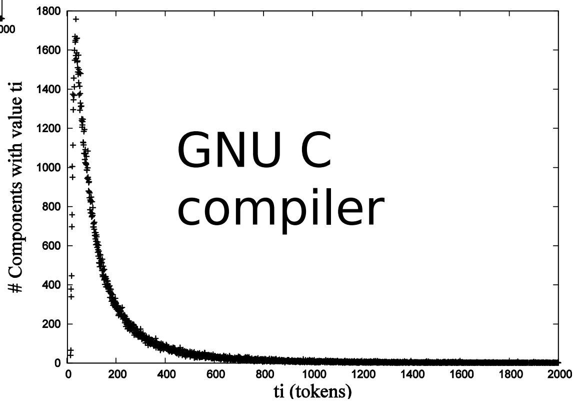


All data from TrEMBL genomic databases.

Scale invariance - software







All data from Open source downloads

Coding and meaningless symbols: Emergent properties

- Just like Zipf and books, *whatever language you choose and whatever your program does, your system as a whole is overwhelmingly likely to share the same emergent properties, safety-critical or not.*
- Some of software systems most important properties are mechanism- and meaning-agnostic: *lengths of components, distribution of defects* …

The future of coding in safety-critical systems (reprise)

- There **will** be more languages; and more; and more … because “coding” will remain a fashion concept.
- New languages will not learn from old languages until Computer Science grows up and becomes a legitimate, i.e. falsifiable, science.
- Software systems just like all collections of discrete symbols have important emergent properties which transcend coding and indeed design.

Reference

LH writing site:-

http://www.leshatton.org/

lesh@oakcomp.co.uk