“A linear interpolant model for the pre-dinner speech"

Les Hatton

Professor of Forensic Software Engineering
CISM, Kingston University
L.Hatton@kingston.ac.uk

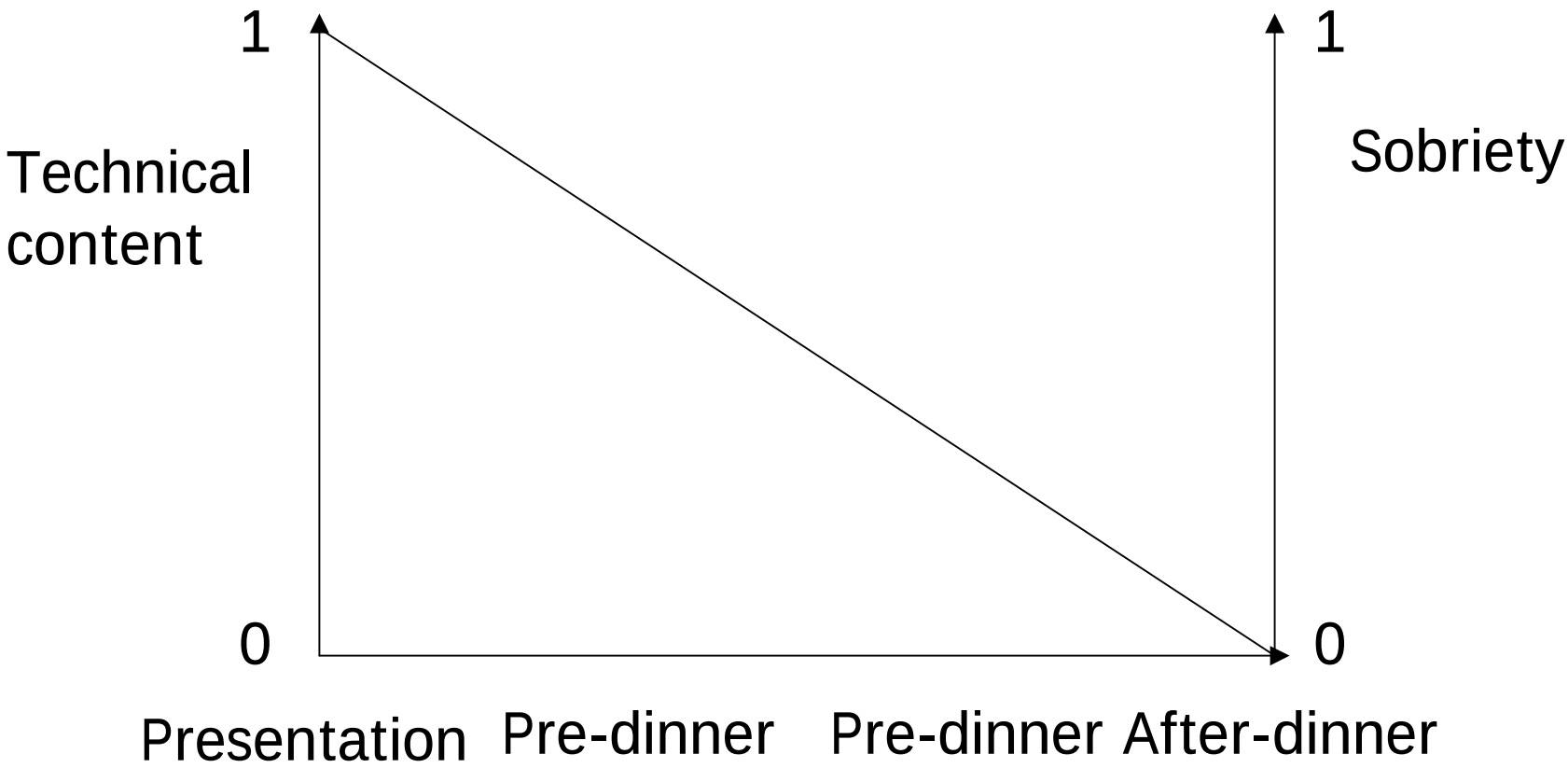
Version 1.1: 16/Jun/2011

A short note on semantics

Note the difference between:

- Once played drums with Alexis Korner
- Played drums with Alexis Korner once.

How a pre-dinner speech might work



So here goes …

- Who cares, software is all the same anyway
- Design or ****-up
- Some safety-critical systems can’t be specified

Consider building a system as follows

Consider a general software system of T tokens divided into M pieces each with t_i tokens, each piece having a *Shannon information content* I_i associated with it. *Note: no nesting.*

1	2	3	….			
			t_i,I_i			
					…	M

$$T = \sum_{i=1}^{M} t_i$$

$$I = \sum_{i=1}^{M} I_i$$

A general theorem for code partitioning

"In any software system irrespective of language or design methodology, conservation of size and information (i.e. choice) is overwhelmingly likely to produce a power-law alphabet probability distribution in the components used to build it", Hatton(2010).

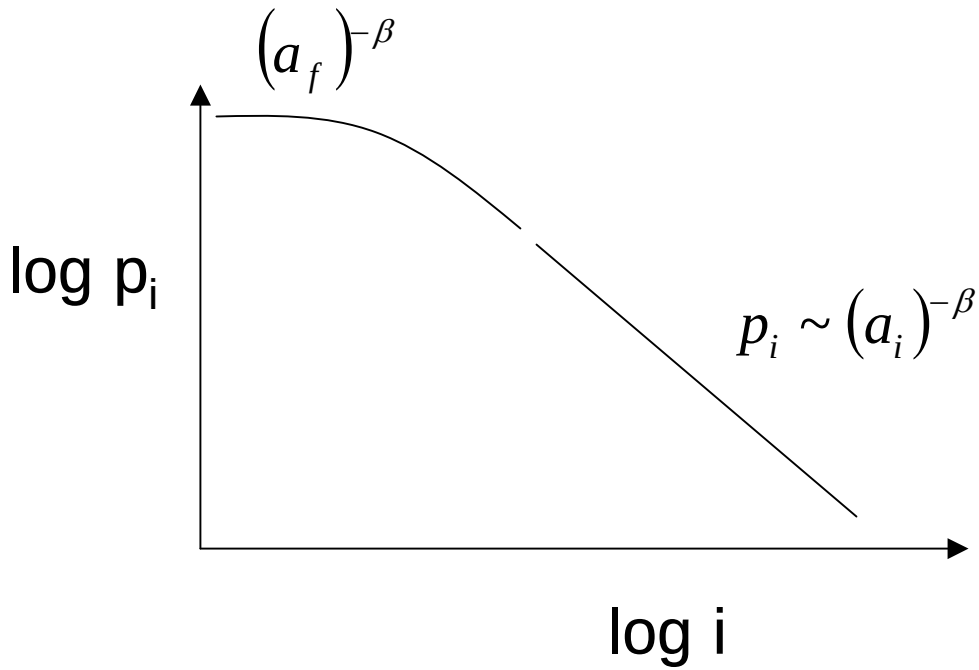
http://www.leshatton.org/variations_2010.html

$$p_i \sim (a_i)^{-\beta} = (a_f + a_v(i))^{-\beta}$$

Fixed Variable

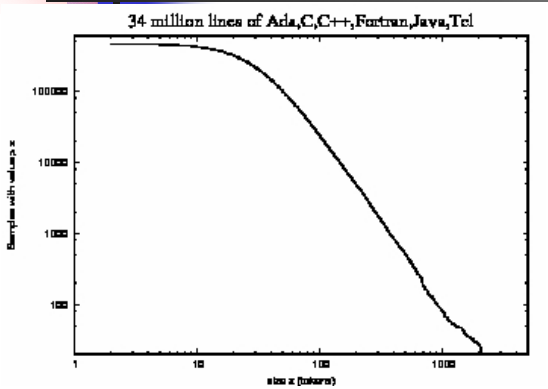
Application to software systems

So we are looking for the following signature

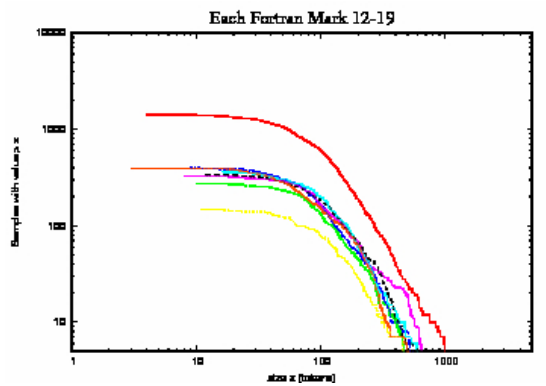


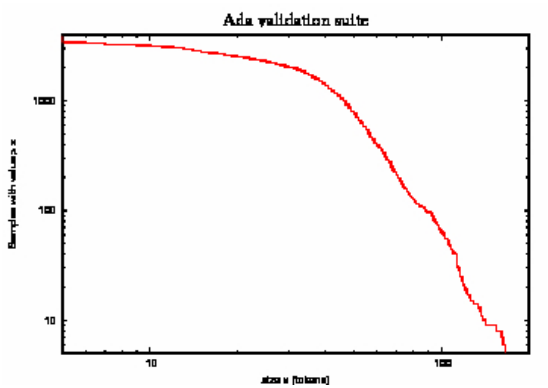
(N.B Requires writing lexers for each language to validate.)

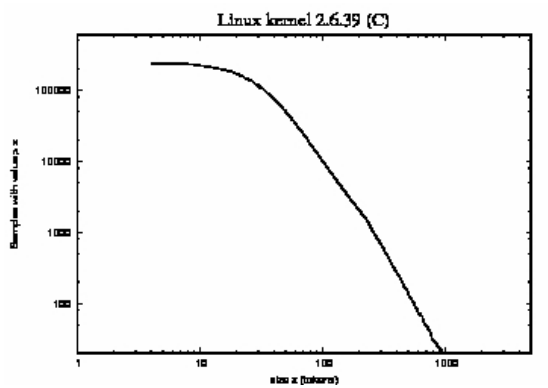
Some results

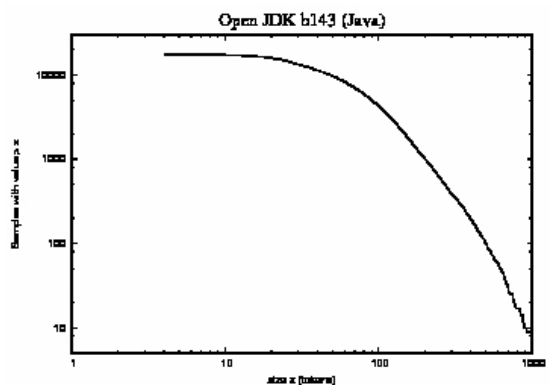


34 million lines of Ada, C, C++, Fortran, Java, Tcl in 74 systems.



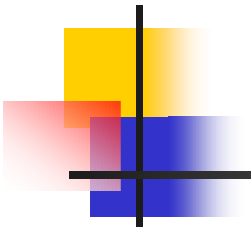






Ada C Java

The “who cares its only software” principle



- Human programmers will always unconsciously subvert any attempt to impose paradigm-based structuring principles.
- Human programmers will distribute tokens independently of the language they are using.

On to the next bit …

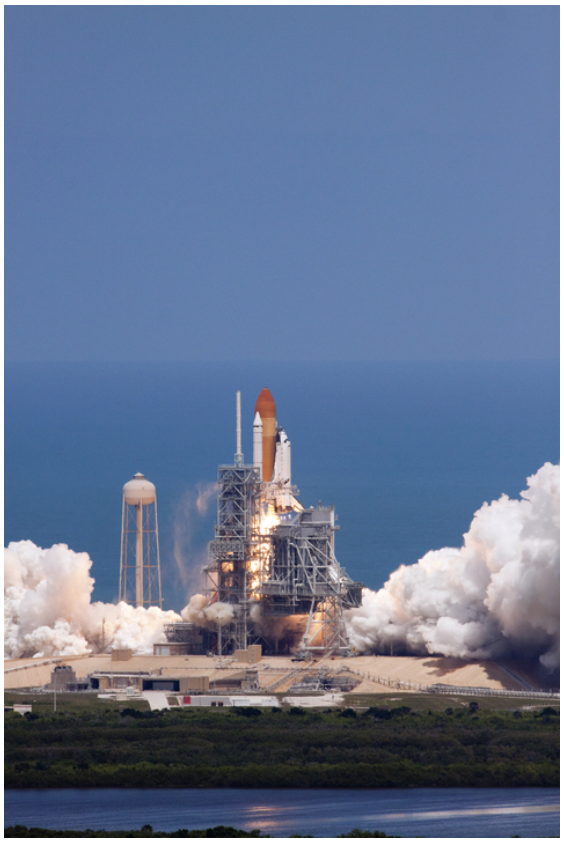
- Who cares, software is all the same anyway
- Design or ****-up
- Some safety-critical systems can't be specified

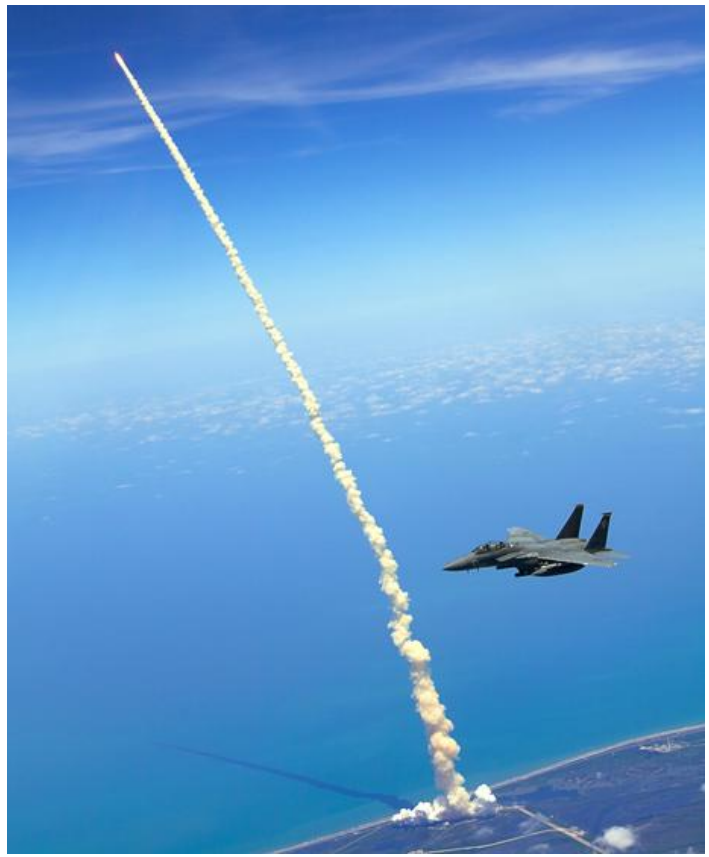
Design or Mash-up

- Specify; Build when understood
  - A paradigm for which Ada is specifically and eminently suitable.
- Agile; Repeatedly hit customer over head with unfinished copies until they surrender.
  - What nearly all software development is about today.

The bottom line in all of this activity is that ...

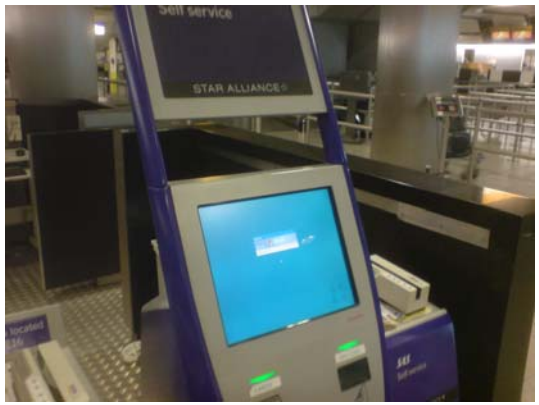
Some of it is exceedingly good …
(Space Shuttle software)





… but most of it is not
(all within 90 minutes at Heathrow, 11-May-2010)

Check -in



Departures

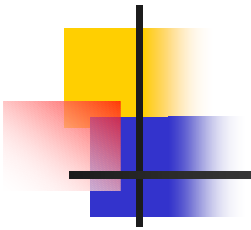
"This system is rubbish"
(departures official)

Departures lounge

On the plane

Honesty is the best policy

So here goes …

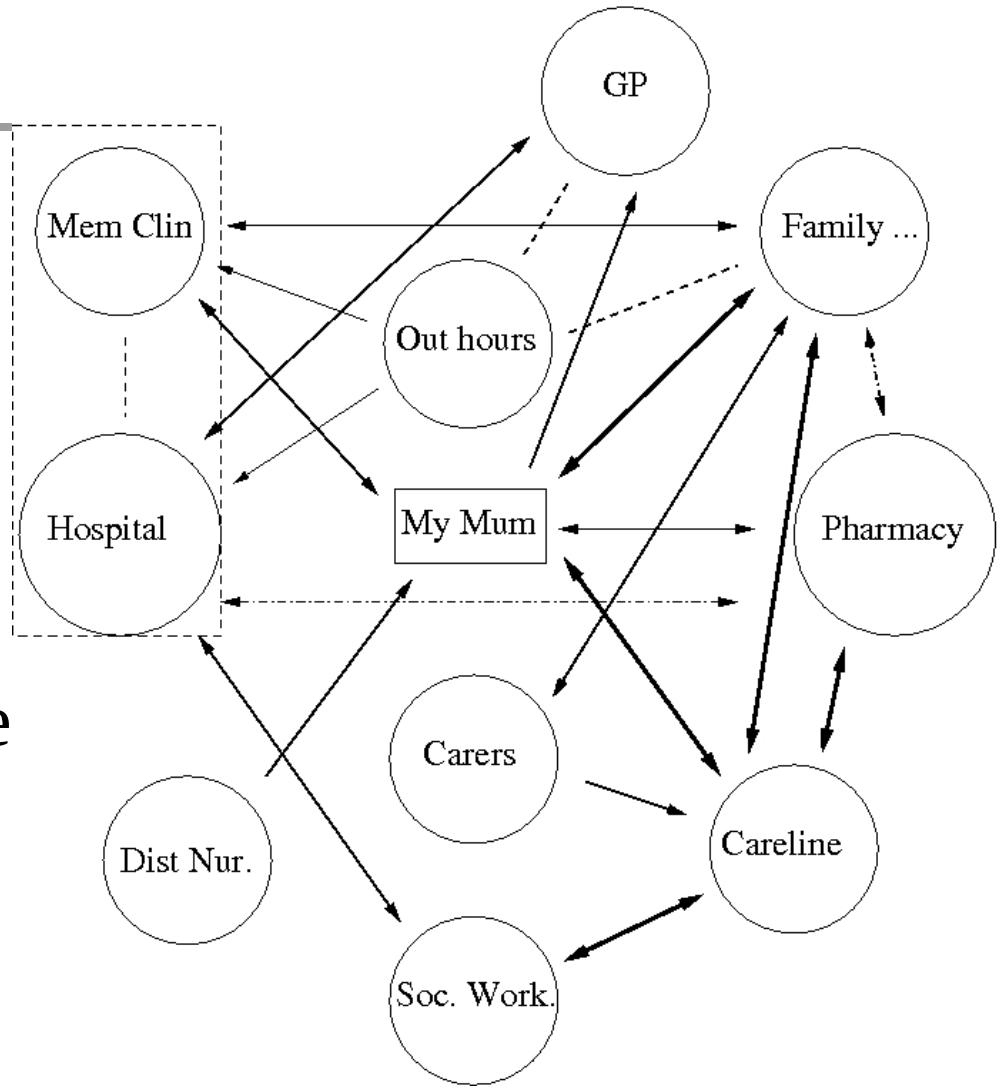


- Who cares, software is all the same anyway
- Design or ****-up
- Some safety-critical systems can’t be specified

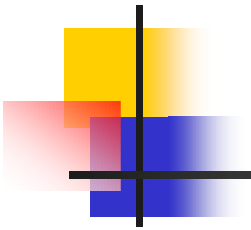
My Mum and the NHS

Rules of engagement

- Nobody is allowed to speak to a doctor
- Communication is frowned upon
- Confidentiality is more important that patient health
- IT is almost irrelevant



Some tentative thoughts



- CS education is mostly producing *Mashers* not *Specifiers* leading to a growing skills gap
- Many important systems are wrecked by the ephemeral nature of their political and social context. It is useless trying to specify them. The essence will be to find *fixed specification points* (s_i such that $R(s_i,t) = s_i$) and ignore everything else.
- In spite of the above, some systems MUST be built by careful specification and this will remain a vital service.

References

My writing site:-

http://www.leshatton.org/

Thanks for your attention.