“The role of empiricism in improving the reliability of future software"

Les Hatton

CISM, Kingston University
L.Hatton@kingston.ac.uk, lesh@oakcomp.co.uk

Version 1.1: 25/Aug/2008

Overview

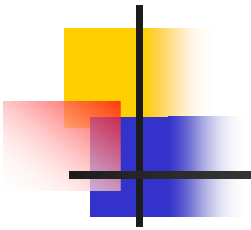
"Good tests kill flawed theories; we remain alive to guess again"

"Science must begin with myths, and with the criticism of myths"

Karl Popper 1902-1994

Copyright Les Hatton, 2008-

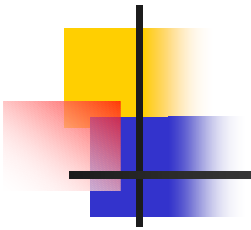
Overview



- Some history
- Software metrics, the bad, the worse and the ugly
- Scale-free behaviour and statistical mechanics
- Some more useful empirical results

Copyright Les Hatton, 2008-

Fashion … languages, bloody languages



In my career, I have been forced to write programs in:-

- Focal
- Atlas Autocode
- Algol
- Various assemblers, FPS microcode, MVS JCL, SEL firmware
- Fortran 66, 77, 90
- C
- Pascal, Occam
- Ada (briefly)
- C++, Java 1 and 2
- Various scripting languages, PHP, Perl, Tcl/Tk, Bash, Javascript, MySQL, …
- and back to C again, (this time from choice)

Copyright Les Hatton, 2008-

Fashion … incomprehensible tomes

Language	Typical number of pages
Java	1,200
PHP	900
Javascript (Have fun with …)	1,000
C++ standard	808
C++ book	1,400
C book	250
My beloved complete Feynman Lectures on Physics	1,500

Copyright Les Hatton, 2008-

Fashion … paradigms, bloody paradigms

In the final year at Kingston 2008, third year projects used

- C, C#, C++, Java, Perl, PHP, MySQL, XML, HTML, XHTML, VB.Net on XP, Mac OS X, Linux, Vista with Eclipse, Netbeans, Ant, JWSDP, Glassfish, Linden script, DreamWeaver, Flash, Developer Studio.Net and a few others I can't recall.

This is to satisfy the *needs of industry* as if they had any idea what they wanted.

Copyright Les Hatton, 2008-

Fashion … inadequate statistical reasoning

- Very few papers attempt to establish significance for their results using standard methods
- A typical result might read *A > B therefore A is (better / worse – substitute as appropriate) than B.*

Copyright Les Hatton, 2008-

And the result …

In the USA …

- 01/08/2008 US Office of Management and Budget have identified approximately 413 Government projects totalling at least $25 billion which are poorly planned, poorly performing or both.
  - http://blogs.spectrum.ieee.org/riskfactor/2008/08/

Copyright Les Hatton, 2008-

and …

A small selection from the UK:-

- 25/07/2008 BBC Radio Humberside system failure delays radio transmissions.
- 31/07/2006 Software failure hits 80 National Health Service Trusts
- 20/08/2006 Failure in software caused Western England homes to run out of water.
- 20/10/2005 Another Post Office systems failure delayed post
- 03/05/2005 Systems failure delayed ambulances in Western England
- 12/04/2005 Child Support Agency in crisis because of problems with new system
- 11/03/2005 Banking failures invalidate PIN cards
- 2000 (twice), 2002 (twice), 2004 and 2005. Air traffic control systems failures in South East England leading to many delays. The 2004 one took the whole country out.
- 22/12/2004 Post Office systems delayed pension payments
- 23/08/2004 Benefit system (again)
- 06/09/2003 Software failure disrupts British Airways flights world-wide
- 17/10/2001 More than 60% of records on Police Criminal Record Check found to be inaccurate
- 20/03/2001 GBP77m immigration service computer system scrapped
- 25/01/1999. Benefit system chaos due to software failure.

Copyright Les Hatton, 2008-

And a day in the life of a mail server (26-Aug-2008) …

Event	# occurrences
Received messages	47,267
Rejected as non-compliant	- 32,501
Silently discarded as junk	- 13,722
Content filtered spam / scam	- 924
Trojans	- 22
Delivered (to 6 domains)	98 (0.2%)
Root dictionary attacks	160

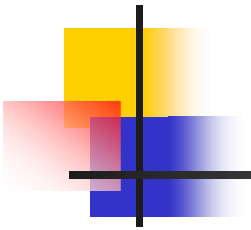
Copyright Les Hatton, 2008-

Some experimental progress

- Tichy (1998) “Should computer scientists experiment more ?”
- The work of Vic Basili, Shari Pfleeger and collaborators
- Journal of Empirical Software Engineering (2002-) is an important step forward

Copyright Les Hatton, 2008-

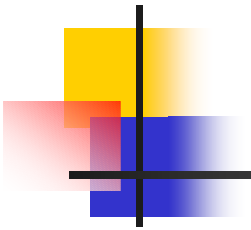
Overview



- Some history
- Software metrics, the bad, the worse and the ugly
- Scale-free behaviour and statistical mechanics
- Some more useful empirical results

Copyright Les Hatton, 2008-

Some examples of ‘metrics’



- The *goto* statement
  - Naur (1963), Dijkstra (1968) and hundreds of others
  - Prohibited in Simula, Occam
  - Banned by MISRA C (1998,2004), ESA Ada (1998), JSF C++ (2005),
- if … else if … with no else, (MISRA, JSF)
- Maximum depth of control nesting, (ESA)
- Maximum cyclomatic number, (most use this)
- Perhaps 40 or so in general use.

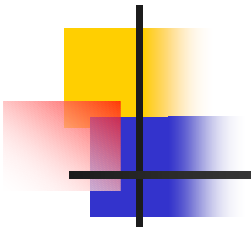
Copyright Les Hatton, 2008-

Metrics
Where do we start ?

- Need a project which:-
  - Is large to permit statistical significance
  - Mature so that effects will have appeared if present
  - Well documented defect history
  - Availability of source code through entire life-cycle including all updates
  - Well-specified subject area so that defect definition is as precise as possible.

Copyright Les Hatton, 2008-

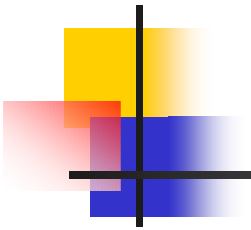
Metrics
Where do we start ?



- Candidate 1: NAG Fortran Library 1970-2000
  - 266,123 XLOC in 3,659 subroutines
  - Development history 1970-1999
  - All defect data embedded in code
  - Analysed source from 13 out of 19 Marks
  - Scientific subroutine library with excellent specs.
- Candidate 2: NAG C library, 1990-1999
  - Developed from Fortran and with similar specs

Copyright Les Hatton, 2008-

Metrics
Where do we start ?

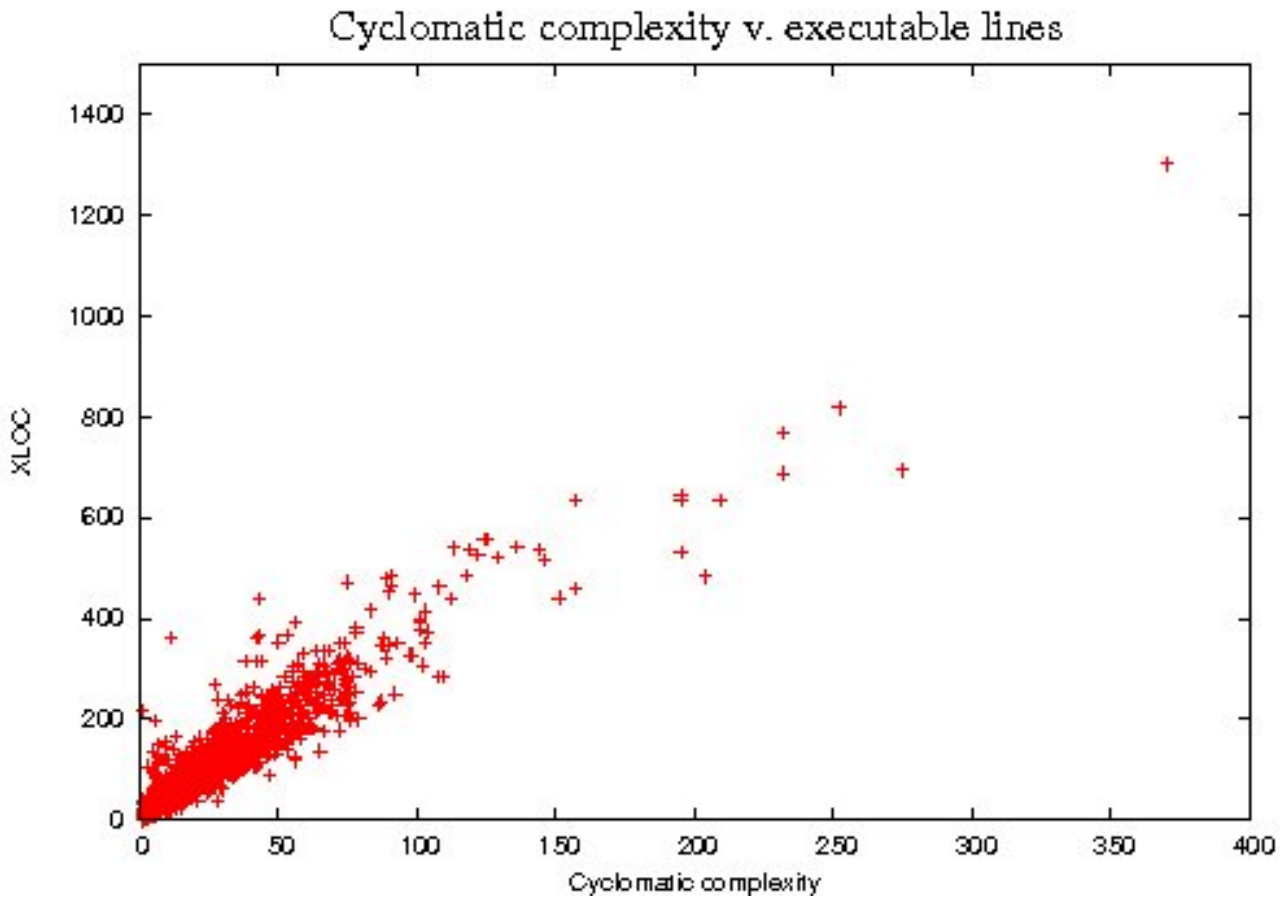


- Candidate 1: NAG Fortran Library 1970-2000
  - Full parse of Fortran, extracted 19 metrics
- Candidate 2: NAG C library, 1990-1999
  - Full parse of C, extracted 7 metrics
- Linear regression, then Principle Component Analysis then smoothing to find any patterns

Copyright Les Hatton, 2008-

Lesson 1:
Cyclomatic complexity effectively useless

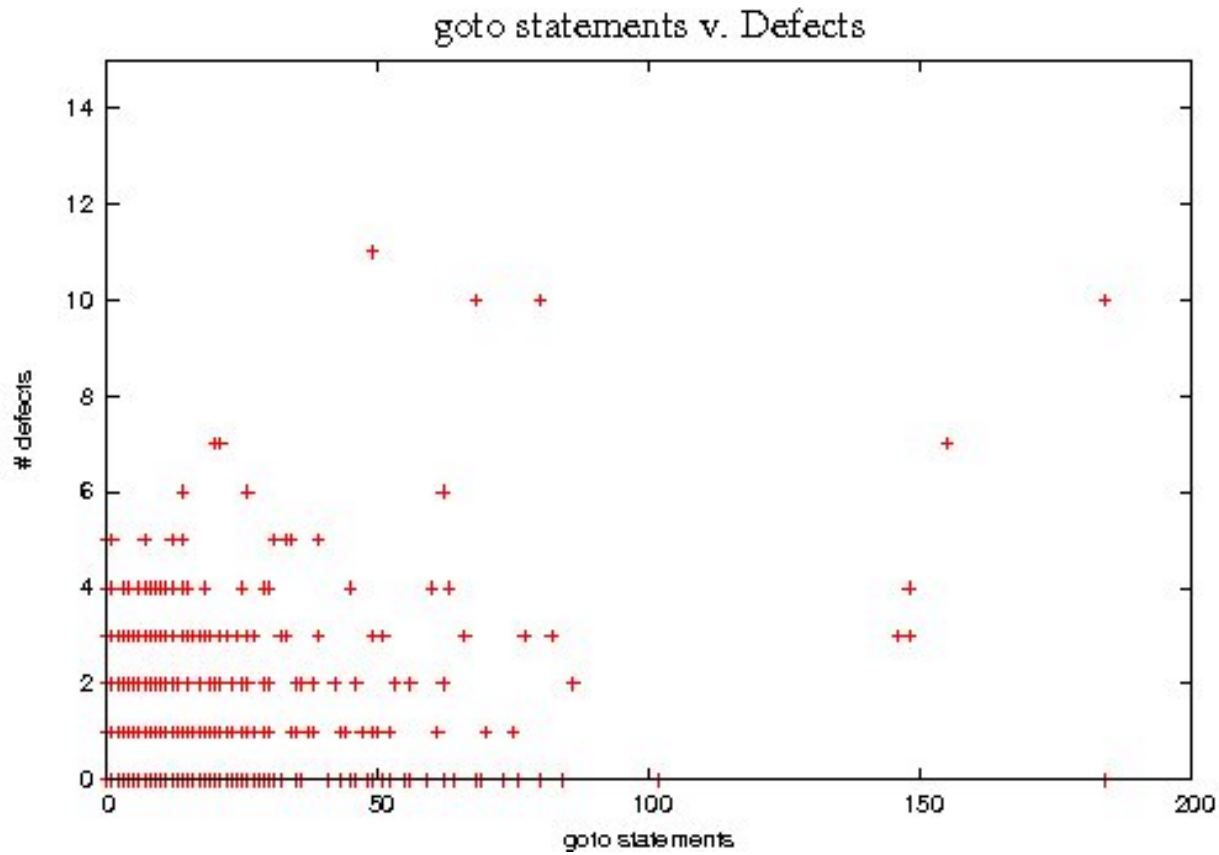
See also:- Woodward et al (1979) (and van der Meulen (2008) on a very large sample).



Copyright Les Hatton, 2008-

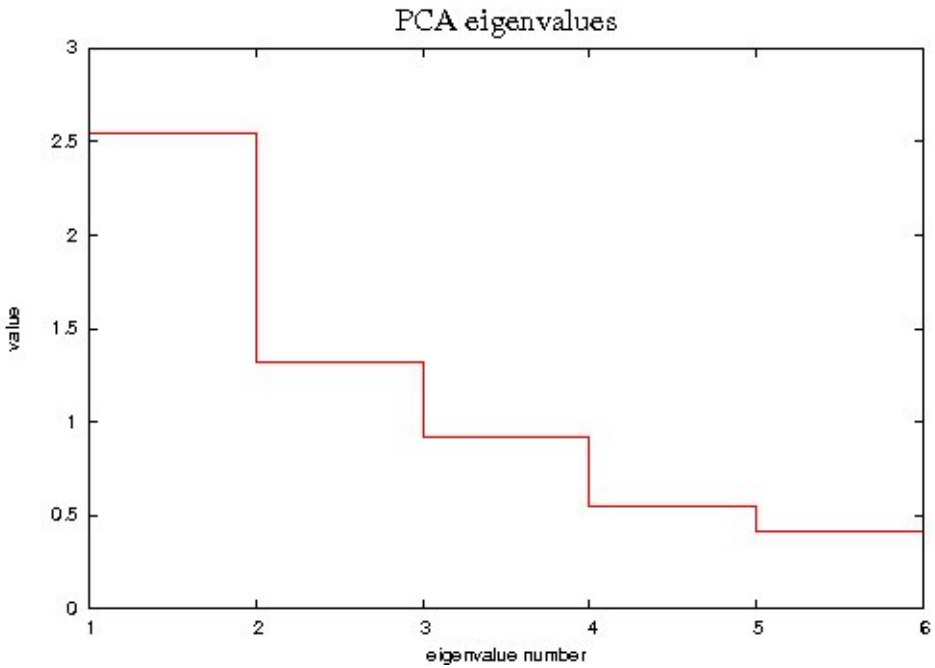
Lesson 2:
No metric strongly correlated

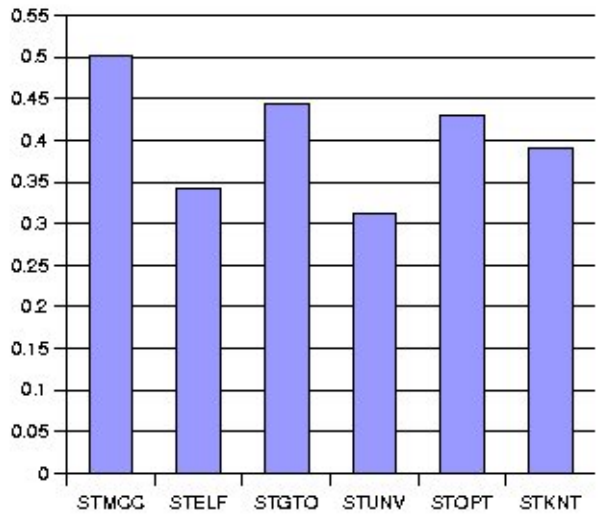
This is typical ...



Copyright Les Hatton, 2008-

Lesson 3:
PCA – a very confused picture

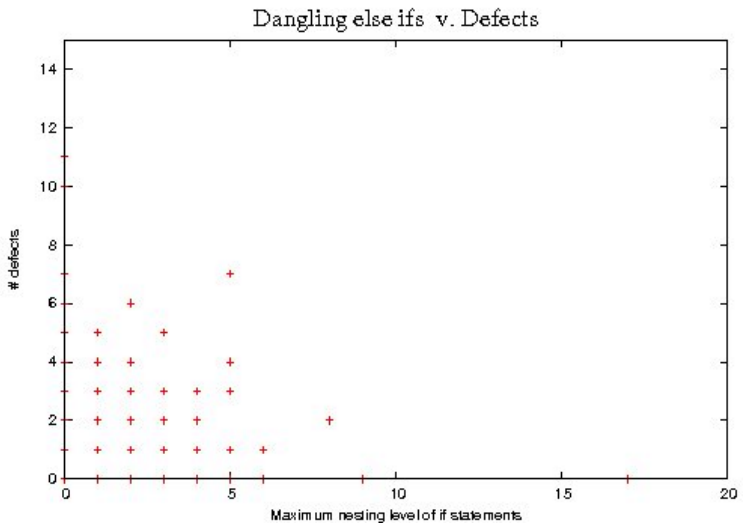


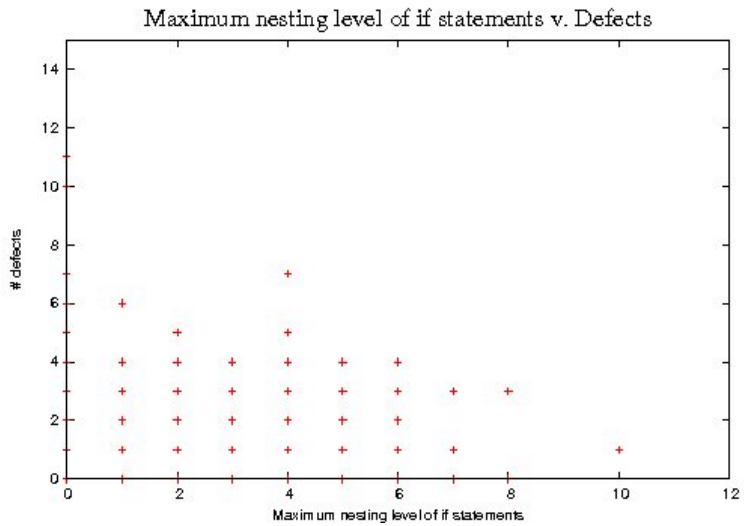


Principle eigenvector

Copyright Les Hatton, 2008-

Lesson 4:
Some metrics weakly anti-correlated



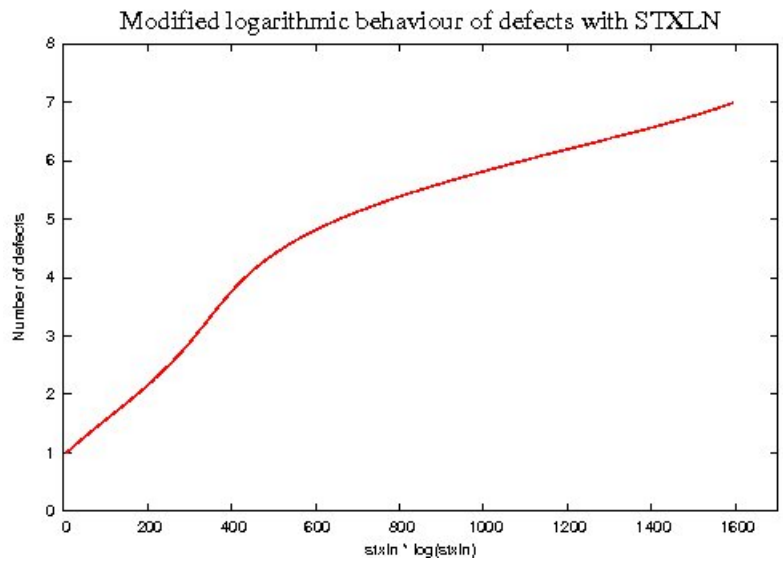


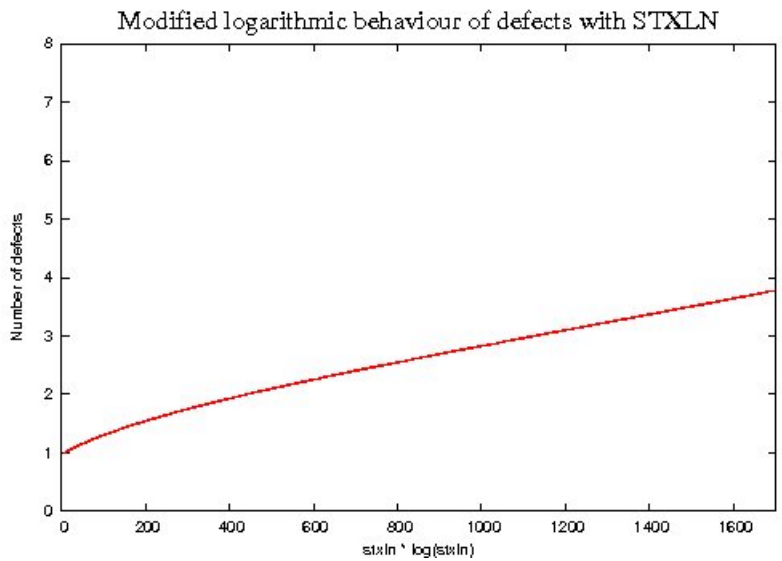
Both are significant at the 5% level and behave opposite to the accepted doctrine

Copyright Les Hatton, 2008-

However, Lesson 5:

With smoothing, there is clear evidence that the number of defects d ~ x ln x where x is the number of executable lines of code.

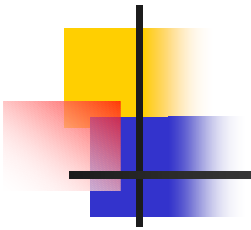




NAG Fortran and NAG C, Hopkins and Hatton (2008). See also Lipow (1982)

Copyright Les Hatton, 2008-

Overview



- Some history
- Software metrics, the bad, the worse and the ugly
- Scale-free behaviour and statistical mechanics
- Some more useful empirical results

Copyright Les Hatton, 2008-

What is scale-free behaviour ?

- In this context, scale-free behaviour refers to a phenomenon whose frequency of occurrence is given by a power-law.
- Consider word-counting in a document. If n is the total number of words in a document and n_i is the number of occurrences of word i, then ***it is observed*** (originally by Zipf (1949)), that for many texts,

$$f_i = \frac{c}{i^p}$$ where c, p are constants and $$f_i \equiv \frac{n_i}{n}$$

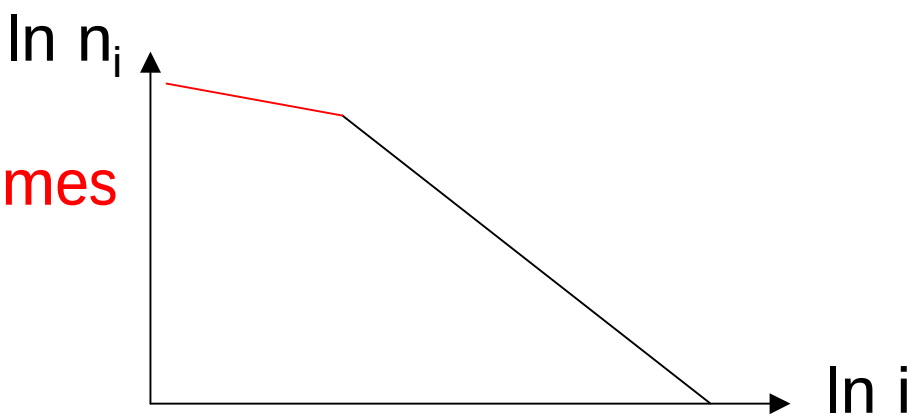
What is scale-free behaviour ?

Re-writing as $$n_i = \frac{nc}{i^p}$$

This is usually shown as

$$\ln n_i = \ln(nc) - p \ln i$$

which looks like



and sometimes

Copyright Les Hatton, 2008-

What is scale-free behaviour ?

For systems with this behaviour we can predict the total length from their ‘vocabulary’.

Summing and re-arranging $n_i = \frac{nc}{i^p}$ for p = 1

Gives

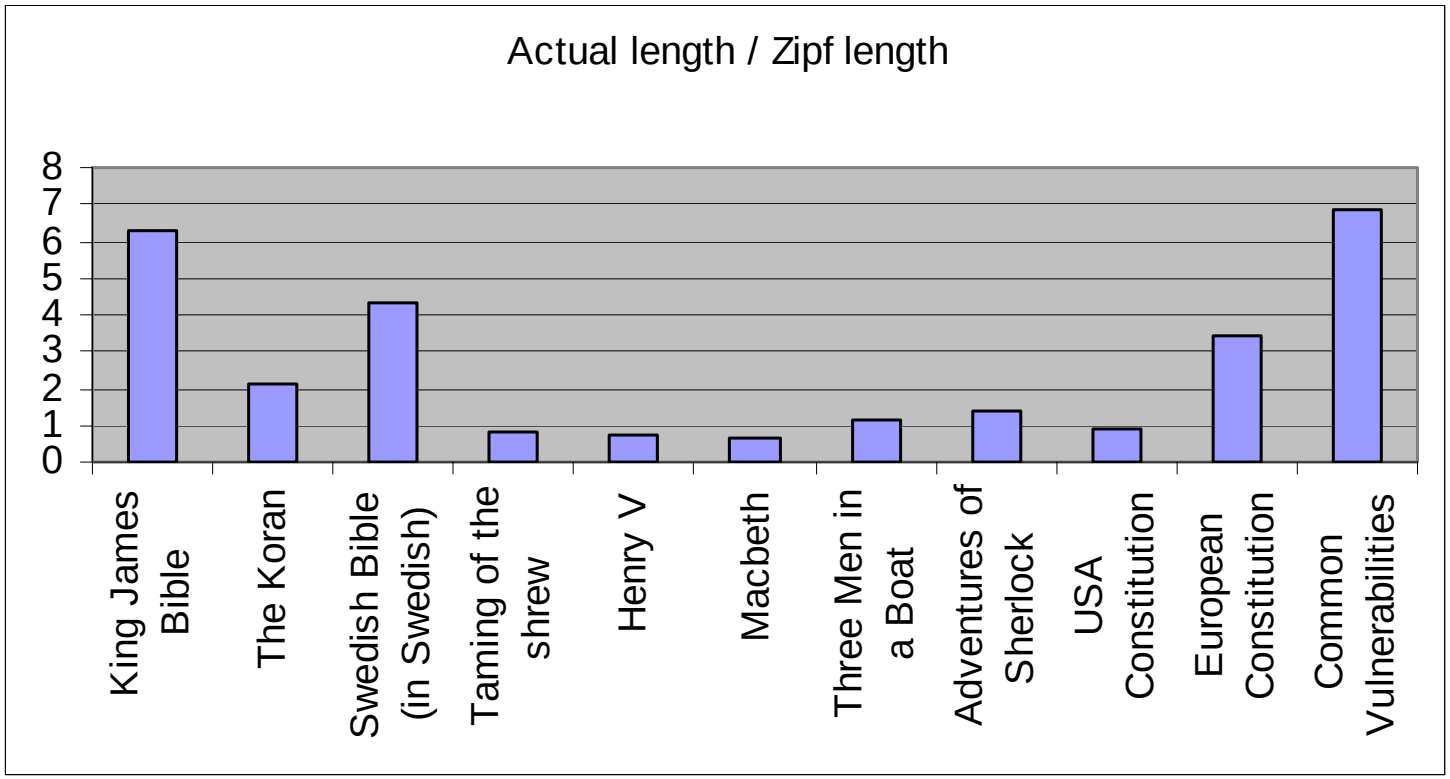
$$n = t(0.5772 + \ln t + O(\frac{1}{t^2}))$$

where n is the total number of words and t is the total number of distinct words

Copyright Les Hatton, 2008-

What is scale-free behaviour ?

Written texts ...

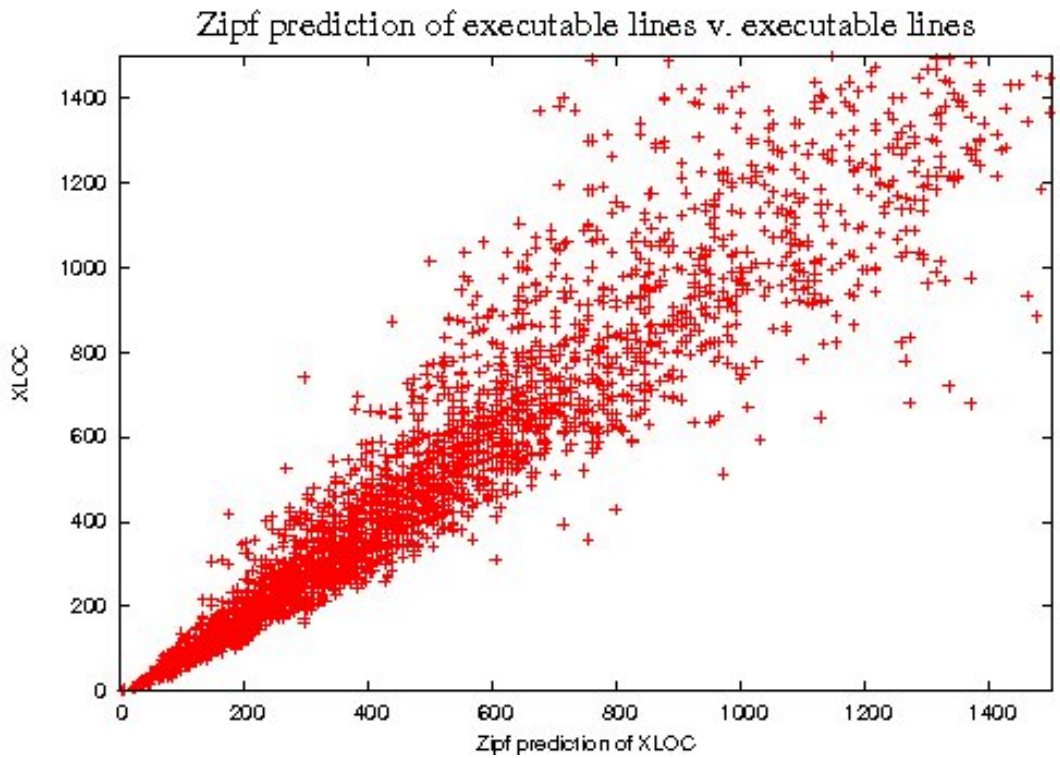


Copyright Les Hatton, 2008-

What is scale-free behaviour ?

NAG Fortran library, (Hopkins and Hatton 2008) ...

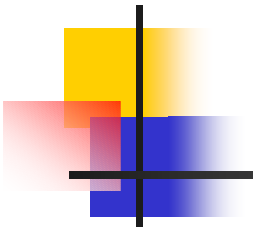
Actual length



Predicted length

Copyright Les Hatton, 2008-

More examples



- Physics:- specific heat of spin glasses at low temperature, Caudron et al (1981)
- Biology: Protein family and fold occurrence in genomes, Qian et al. (2001)
- Biology: Evolutionary models, Fenser et al (2005)
- Economics: Income distributions, Rawlings et al (2004)
- Software systems: incoming and outgoing references and class sizes in OO systems, Potanin et al (2002)
- Fractals also exhibit scale-free behaviour (Novak):-
  - http://cism.kingston.ac.uk/people/details.php?AuthorID=577
- Studies of C systems also reveal scale-free behaviour (Jones)
  - http://www.knosof.co.uk/cbook/cbook.html

Copyright Les Hatton, 2008-

General mathematical treatment

Consider a general system of N atomic objects divided into M pieces each with n_i objects, each piece having a property e_i associated with it.

1	2	3	….			
			n_r,e_r			
					…	M

$$N = \sum_{i=1}^{M} n_i$$

Copyright Les Hatton, 2008-

General mathematical treatment

The number of ways of organising this is:- $$W = \frac{N!}{n_1! n_2! \ldots n_M!}$$

Stirling's approximation + logs as usual gives:-

$$\ln W = N \ln N - \sum_{i=1}^{M} n_i \ln n_i$$

In physical systems, we seek to find the most likely arrangement by maximising this subject to two constraints

$$N = \sum_{i=1}^{M} n_i \qquad \text{and} \qquad U = \sum_{i=1}^{M} n_i e_i$$

Copyright Les Hatton, 2008-

General mathematical treatment

Using Lagrange multipliers and setting $\delta(\ln W) = 0$

leads eventually to the most likely (i.e. equilibrium) distribution being given by

$$p_i \equiv \frac{n_i}{N} = \frac{e^{-\beta e_i}}{\sum_{i=1}^{M} e^{-\beta e_i}}$$

where p_i is the probability of piece i getting a share e_i of U and β is a constant.

Copyright Les Hatton, 2008-

General mathematical treatment

To summarise

The equilibrium distribution of the e_i subject to the constraints

$$N = \sum_{i=1}^{M} n_i \qquad \text{and} \qquad U = \sum_{i=1}^{M} n_i e_i$$

is

$$p_i \equiv \frac{n_i}{N} = \frac{e^{-\beta e_i}}{\sum_{i=1}^{M} e^{-\beta e_i}}$$

Copyright Les Hatton, 2008-

Application to software systems

If we identify e_i with the defect density (d_i/n_i) in a component, then the *total number of defects in a software system* is given by:-

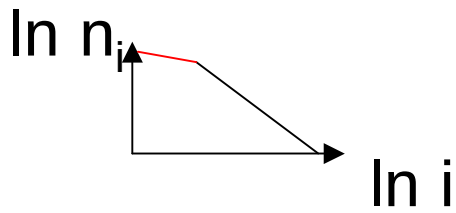
$$U = \sum_{i=1}^{M} n_i e_i$$

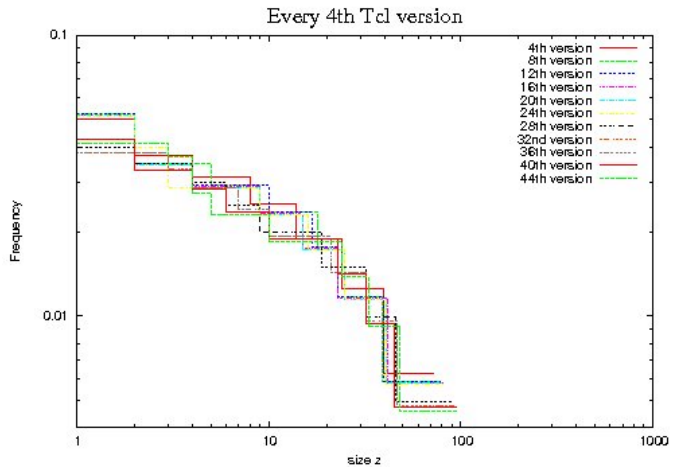
(Note that this only introduces a term $O(1/n_i^2)$ in the Lagrange reduction and is no worse than Stirling's approximation.)

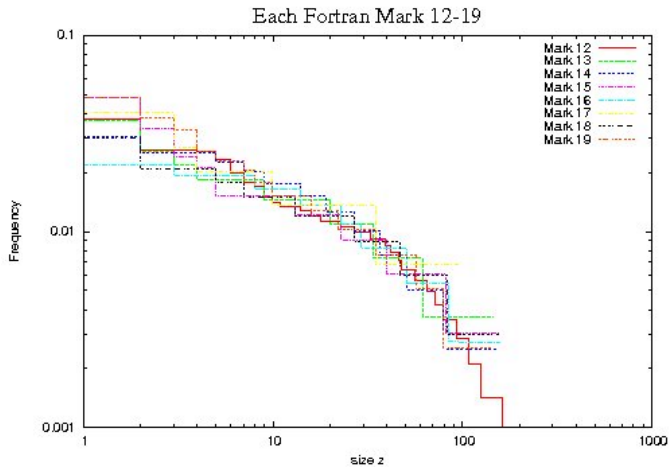
Copyright Les Hatton, 2008-

General mathematical treatment

However, how are p_i distributed in real software systems ?





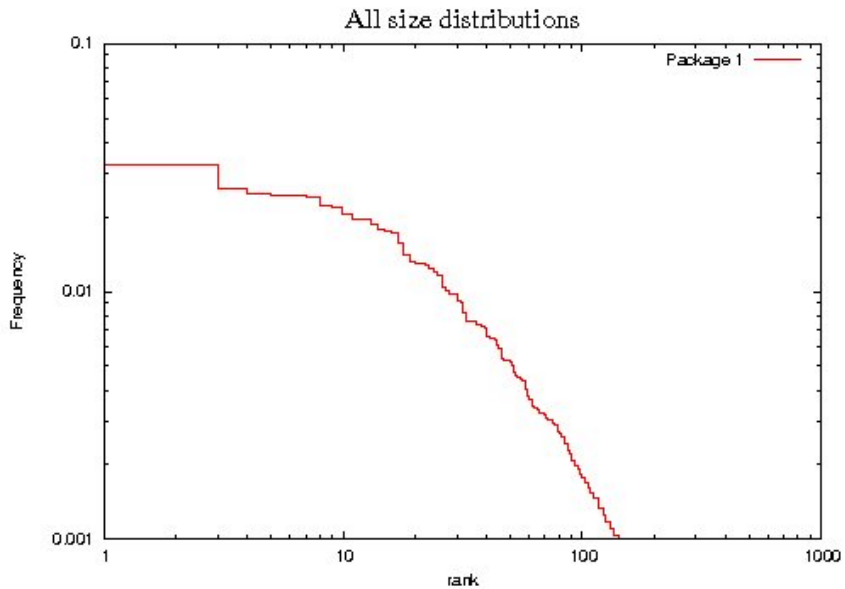


Distribution in two large systems as they mature, i.e. become *quasi-equilibrated*.

Copyright Les Hatton, 2008-

Application to software systems

Averaged over 21 very different quasi-equilibrated systems in Fortran, C and Tcl-Tk

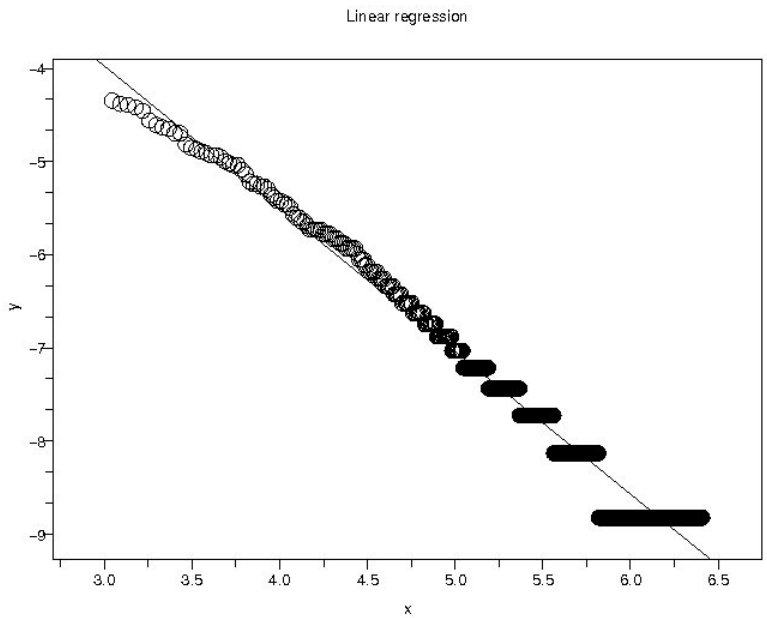


These distributions are classic ***power-law*** behaviour and it appears to be present *a priori*

$$p_i \approx \frac{C}{n_i^{\beta}}$$

Copyright Les Hatton, 2008-

Application to software systems



This is linear with statistical significance at least 1%

Copyright Les Hatton, 2008-

General mathematical treatment

Bringing it all together, the number of defects in a system is effectively frozen at release. So, *as a system approaches quasi-equilibrium* subject to the constraints,

$$N = \sum_{i=1}^{M} n_i \qquad \text{and} \qquad U = \sum_{i=1}^{M} n_i e_i$$

Then a priori ***power-law*** behaviour

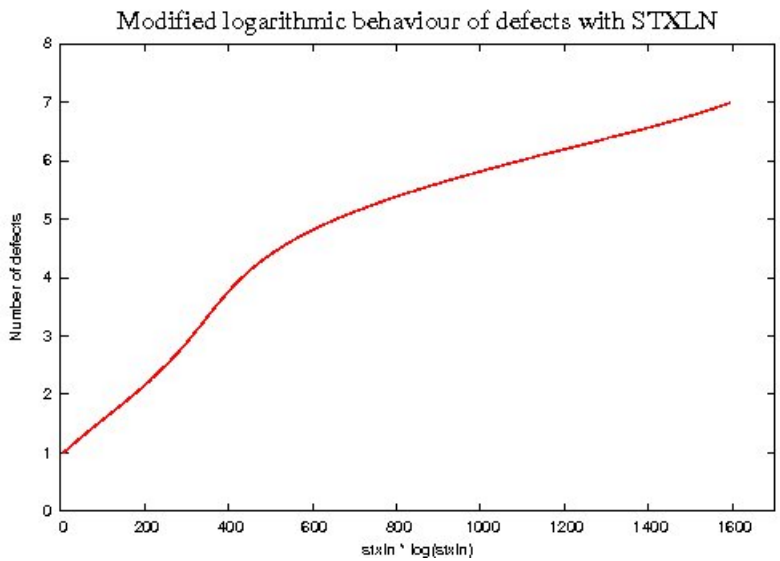
$$p_i \approx \frac{C}{n_i^{\beta}} \quad + \quad p_i \equiv \frac{n_i}{N} = \frac{e^{-\beta e_i}}{\sum_{i=1}^{M} e^{-\beta e_i}}$$

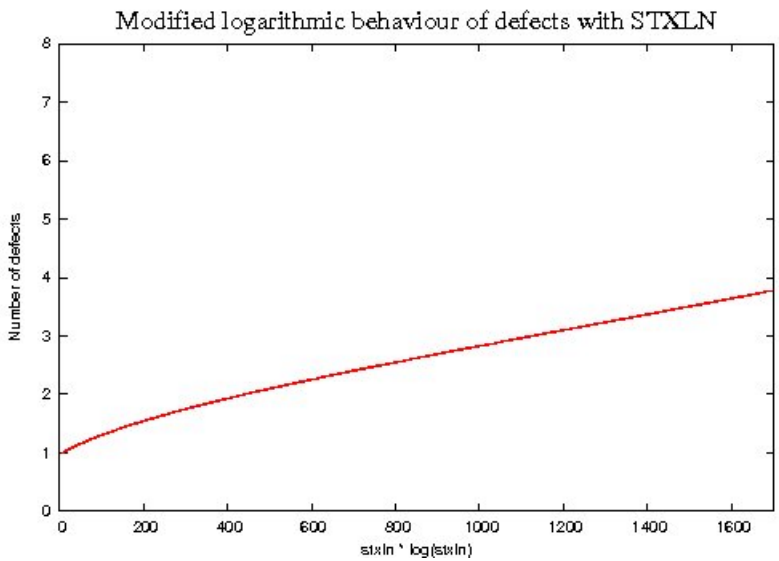
=> $e_i \sim \ln n_i$ or $d_i \sim n_i \ln n_i$

Copyright Les Hatton, 2008-

Application to software systems

and this is what we appear to observe ...

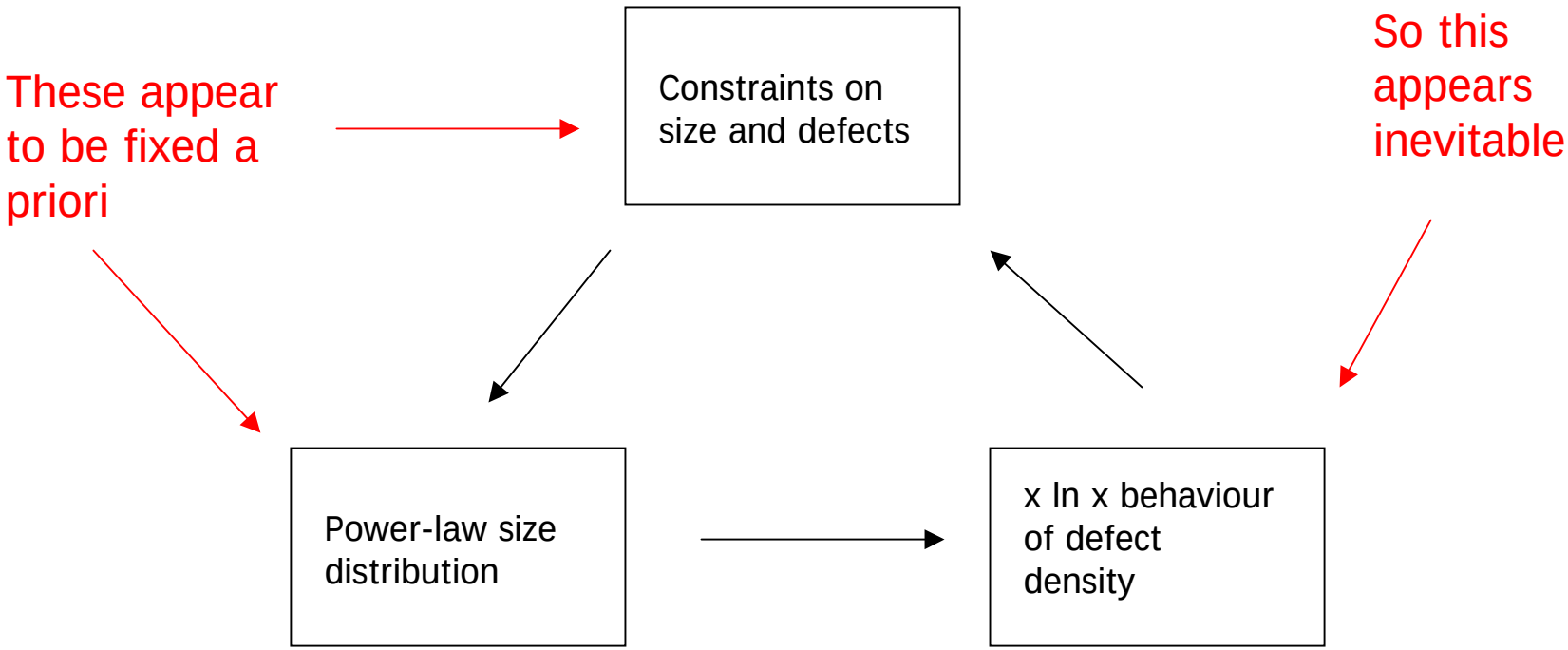




NAG Fortran and C libraries, Hatton and Hopkins (2008). See also Lipow (1982)

Copyright Les Hatton, 2008-

Application to software systems



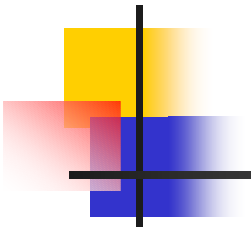
Copyright Les Hatton, 2008-

Application to software systems

Stirling's approximation in the development of the predicted equilibrium state depends on the N and the n_i being large. It has however long been known, (Feynmann and others) that the approximation is surprisingly good in physical systems when these are not so large.

Copyright Les Hatton, 2008-

Overview

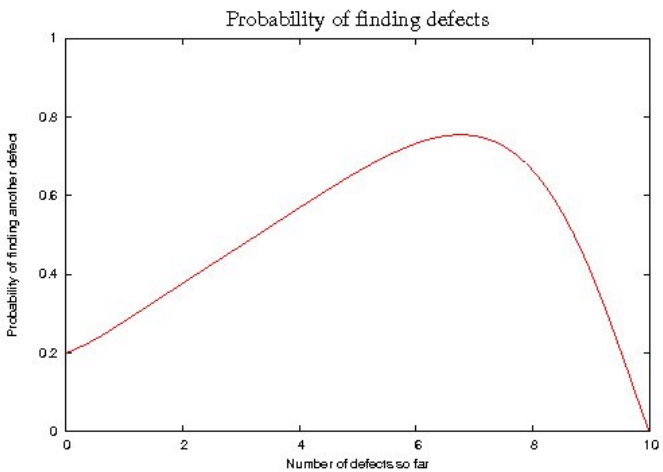


- Some history
- Software metrics, the bad, the worse and the ugly
- Power-law behaviour and statistical mechanics
- Some more useful empirical results

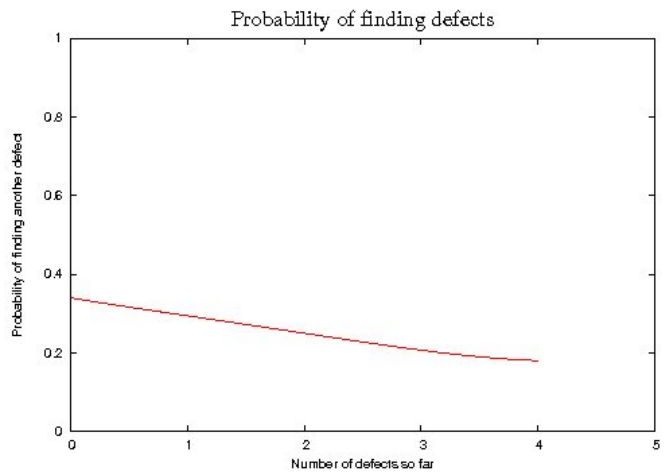
Copyright Les Hatton, 2008-

Some more useful empirical results

- Defects cluster when systems are in quasi-equilibrium (i.e. relatively immature) but not apparently when in equilibrium, (highly significant) (Hatton and Hopkins 2008).



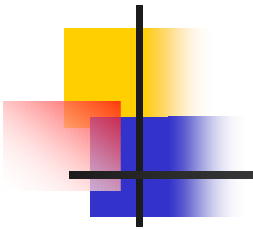
NAG Fortran



NAG C

Copyright Les Hatton, 2008-

Some more useful empirical results

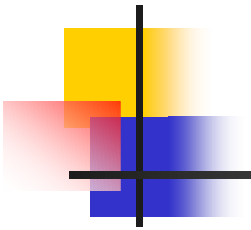


- Note that this clustering is extreme with 78% (2865/3659) of the NAG Fortran components exhibiting no defects
- Even in the C library, 66% (1506/2267) of the components have exhibited no defects

Nothing in either analysis has so far been able to find any reason for this, so the current best guess is that its purely statistical, like asking why a particular person has won the lottery, (the answer being of course that everybody else didn't).

Copyright Les Hatton, 2008-

Some more useful empirical results

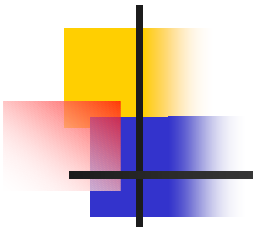


The following are all statistically significant

- Checklists do not appear to improve inspection capability, (highly significant), (Hatton (2008)).
- The Pascal construct for i:= m to n is far more reliable than the C/C++ equivalent for (i=1; i<=m,…) (van der Meulen (2008))
- Run-time checks are just as effective on reliable code as unreliable code so don’t remove them, (van der Meulen (2008)).

Copyright Les Hatton, 2008-

Conclusions



- Empirical results strongly suggest implementation independent behaviour is present in software systems
- Component sizes in software systems of very different size and language obey power-law distributions *a priori*
- Arguments based on statistical mechanics predict defect distribution behaviour which is observed
- Executable lines of code is as good as any other metric
- Individual metrics long-believed to be correlated to defect are weak at best and some appear even weakly anti-correlated.

Copyright Les Hatton, 2008-

Conclusions

- Most importantly, we are beginning to get a feeling for how defective systems behave without being too cluttered with implementation details.

I for one, find this very exciting.

Copyright Les Hatton, 2008-

References

My writing site:-

http://www.leshatton.org/