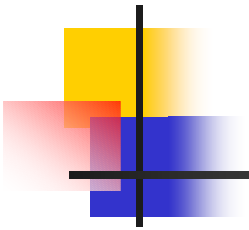
“Conservation of software defect and scale-free behaviour in software systems"

Les Hatton

CISM, Kingston University
L.Hatton@kingston.ac.uk

Version 1.1: 26/Mar/2008

Overview



- What is scale-free behaviour ?
- Examples from the real world
- General mathematical development
- Application to software systems
- Where to go from here ?

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,

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

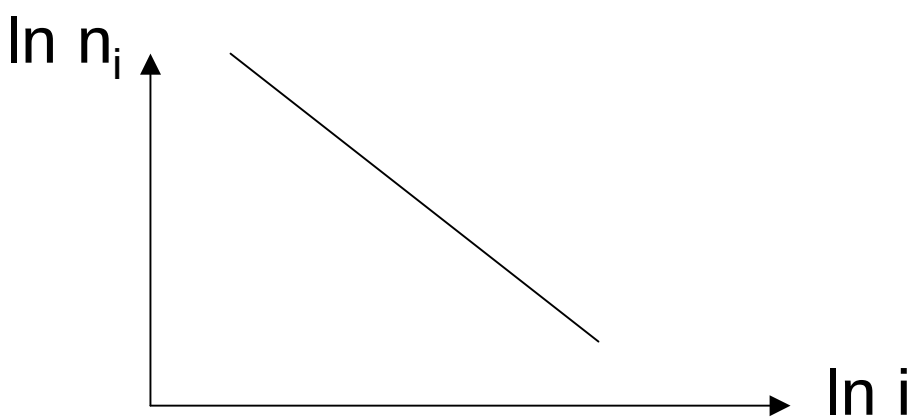
What is scale-free behaviour ?

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



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

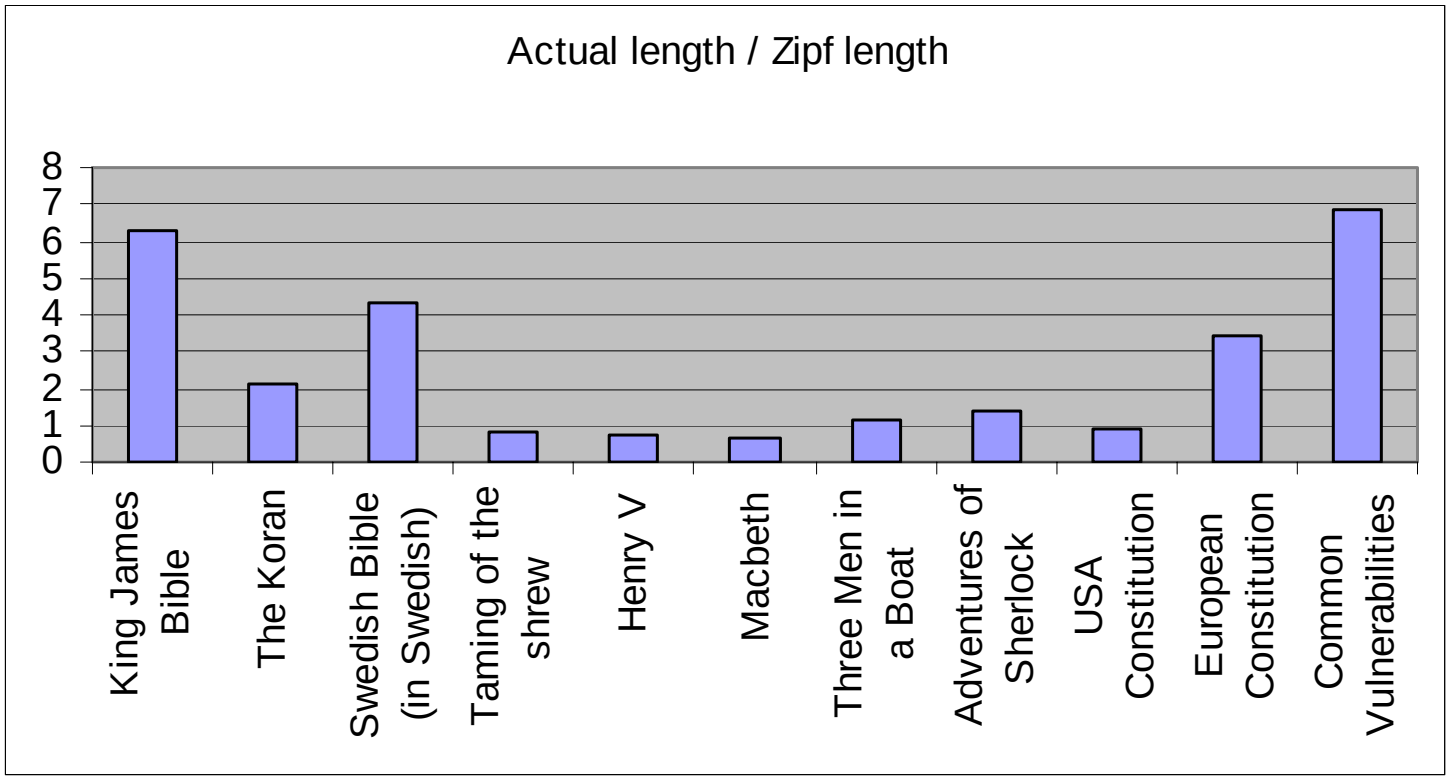
Gives

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

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

What is scale-free behaviour ?

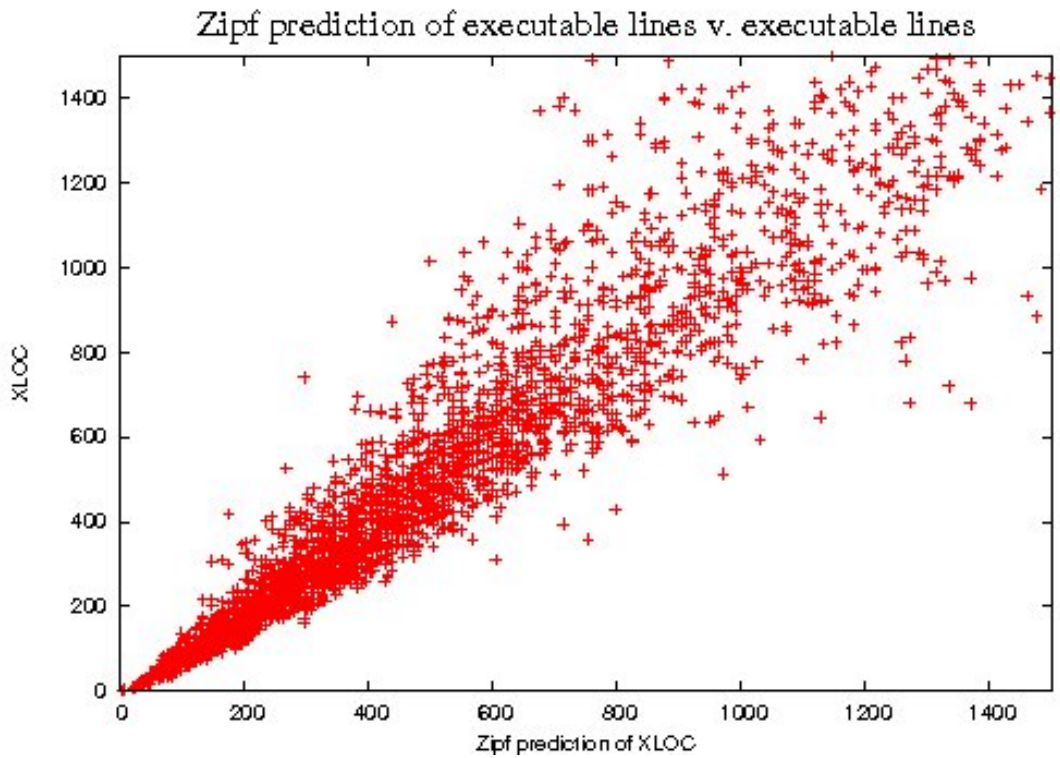
Written texts ...



What is scale-free behaviour ?

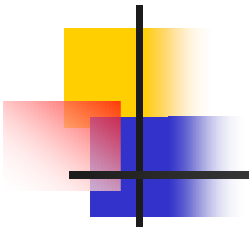
Software library, (Hatton and Hopkins 2008) ...

Actual length



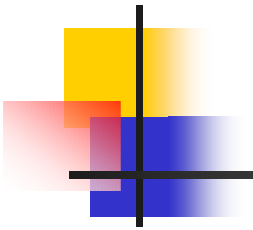
Predicted length

Overview



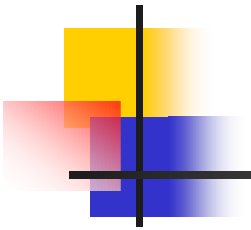
- What is scale-free behaviour ?
- Examples from the real world
- General mathematical development
- Application to software systems
- Where to go from here ?

Examples from the real world



- 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 (Miro):-
  - http://cism.kingston.ac.uk/people/details.php?AuthorID=577
- Studies of C systems also reveal scale-free behaviour (Derek)
  - http://www.knosof.co.uk/cbook/cbook.html

Overview



- What is scale-free behaviour ?
- Examples from the real world
- General mathematical development
- Application to software systems
- Where to go from here ?

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

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

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

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

General mathematical treatment

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

leads to the most likely 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.

General mathematical treatment

To summarise

The most likely distribution of the e_i subject to the constraints

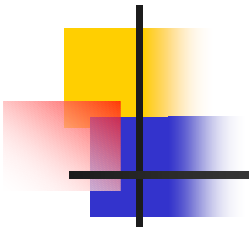
$$N = \sum_{i=1}^{M} n_i \qquad \text{and} \qquad U = \sum_{i=1}^{M} n_i e_i$$

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

but if e_i is logarithmic with n_i, the result is a ***power-law***

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

Overview



- What is scale-free behaviour ?
- Examples from the real world
- General mathematical development
- Application to software systems
- Where to go from here ?

Application to software systems

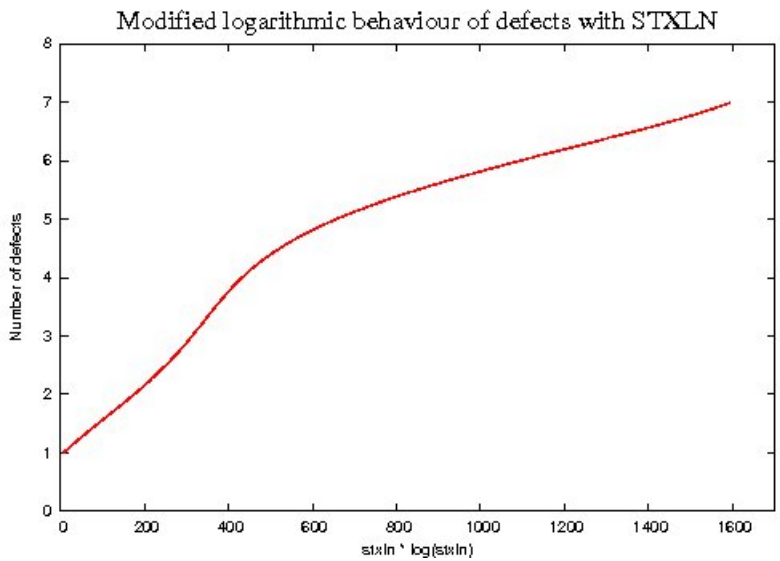
If we identify e_i with the defect density in a component, then the *total number of defects in a software system* is given by:-

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

Application to software systems

However, it has very frequently been observed that the defect density e_i in a component behaves as:-

$$e_i \approx \ln n_i$$



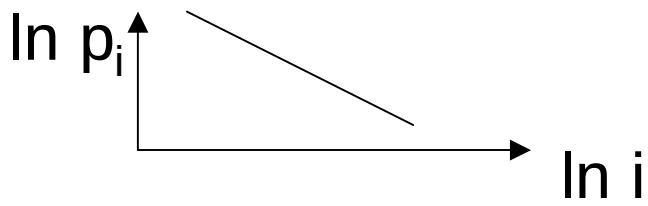
Example from NAG library, Hatton and Hopkins (2008). See also Lipow (1982)

Application to software systems

The implication is that if we design a software system of a certain size, and *the total number of defects is fixed for some reason*, then the most likely system to emerge will have component sizes obeying a power-law distribution.

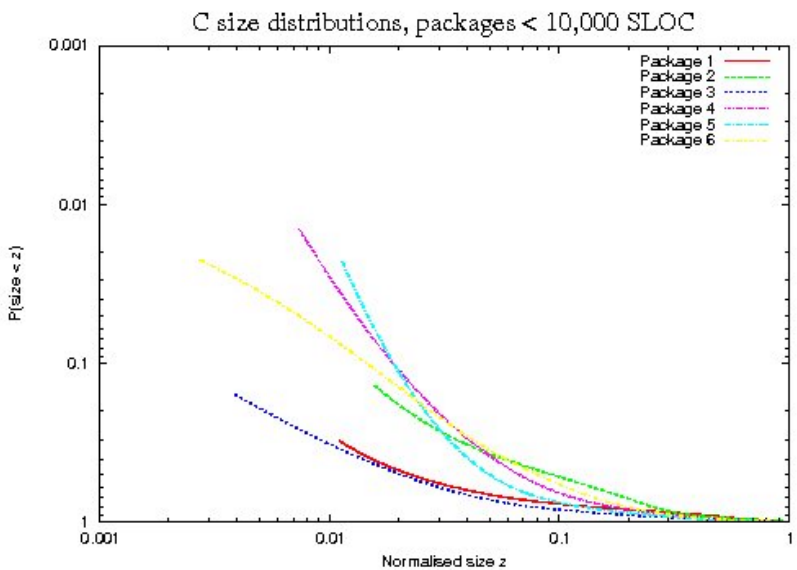
But do we see this ? The following is a study of component sizes in 21 software systems in 3 different languages, Fortran, C and Tcl.

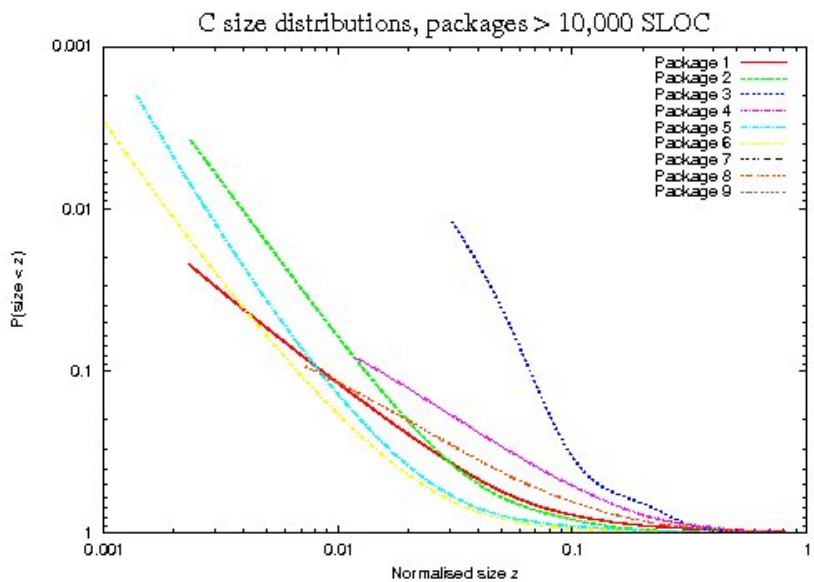
We are looking for .....



Application to software systems

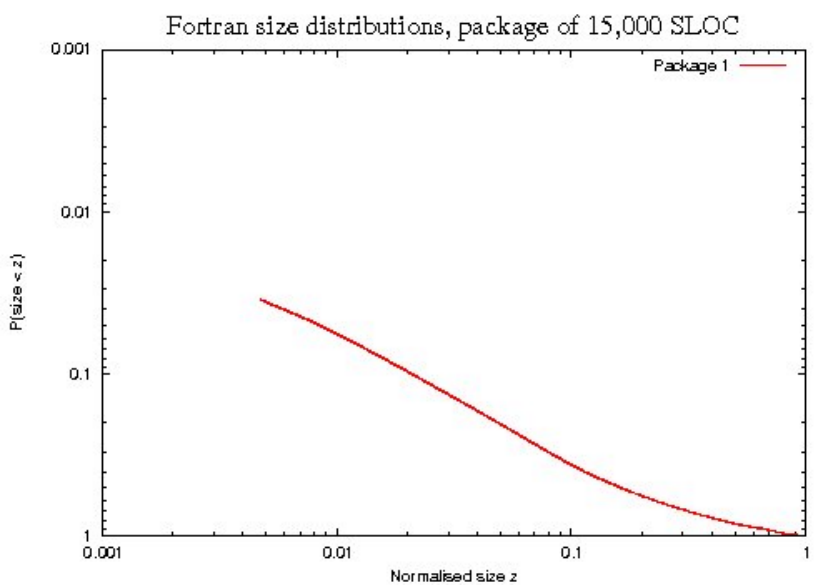
C systems < and > 10,000 LOC

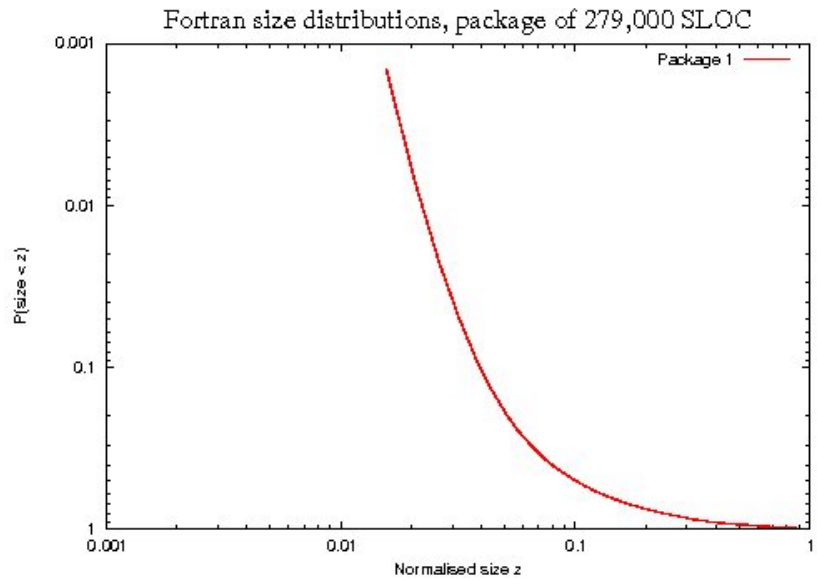




Application to software systems

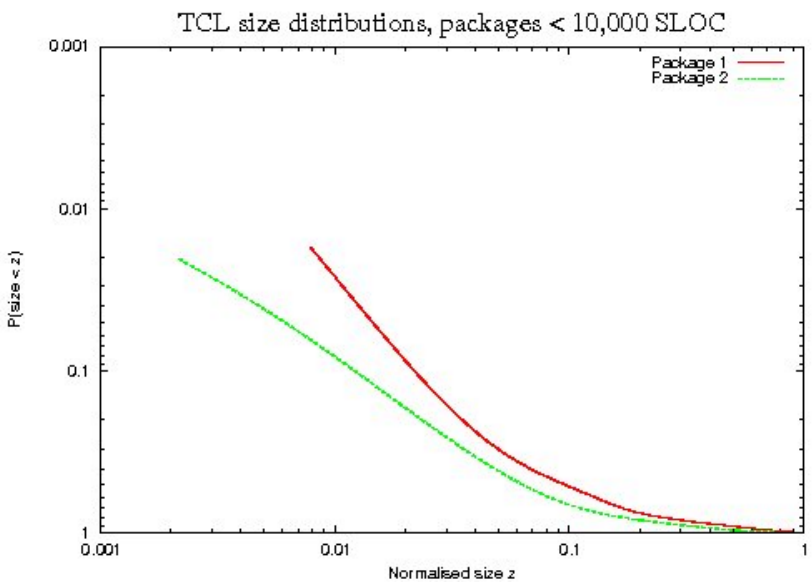
Fortran systems < and > 15,000 LOC

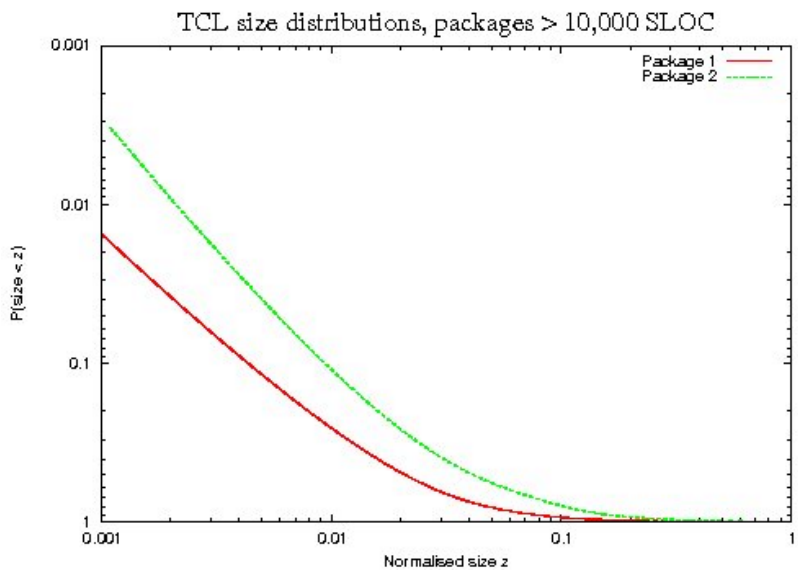




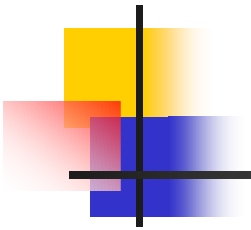
Application to software systems

Tcl/Tk systems < and > 15,000 LOC



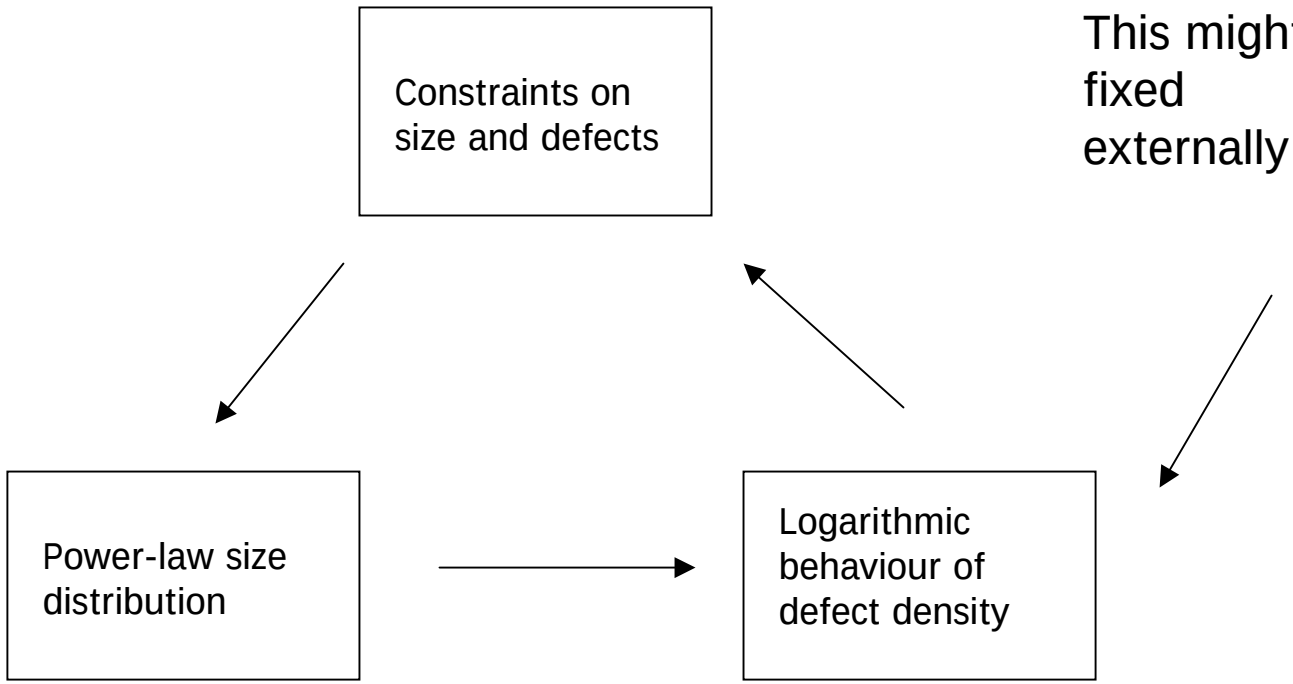


Overview



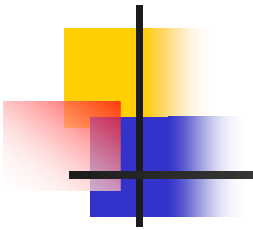
- What is scale-free behaviour ?
- Examples from the real world
- General mathematical development
- Application to software systems
- Where to go from here ?

Application to software systems



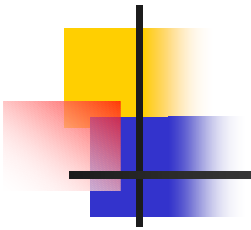
I think I can prove any two imply the third but which is the driver ?

Conclusions



- Component sizes in software systems of very different size and language obey power-law distributions
- Standard arguments from statistical mechanics show that this is inevitable if the total number of defects is approximately conserved.
- The analogy with statistical mechanics can be extended further
- More experiments are necessary to determine the driver.

References



My writing site:-

http://www.leshatton.org/