2000-

"eRisk and eBenefit: the eConomics of software testing"

by

Les Hatton

Oakwood Computing, Surrey, U.K. and
the Computing Laboratory, University of Kent
lesh@oakcomp.co.uk

Version 1.1: 12/Oct/2000

Airbus involved in softwall failure

29/Sep/2000

Indian Airlines Airbus A320 flight IC229 had to circle Gauhati airport several times because an elephant broke through a wall and strolled around the airport.

An observation

We can develop a theory and practice of software testing but when a new software technology appears, many people seem to believe that it no longer applies.

This implies that most people see software testing as a tool or set of tools rather than a methodology.

Overview

- **An introduction to Risk**
- **Examples and sources of risk and failure**
- **Software Risk Mitigation**

Definitions

- **Risk** is when you don't know what will happen but you do know the probabilities
- **Uncertainty** is when you don't even know the probabilities

The eternal conflict

The study of risk is an eternal struggle between:-

- Those who wish to quantify it
- Those who feel it cannot be quantified

A mathematician's view of risk

If R is the Risk, F the Frequency and C the Consequence:

R=FxC

So unlikely catastrophic events have a similar risk to very frequent but unimportant events.

Mathematician's always seek to quantify risk.

A risk practitioner's view of risk

It is fundamentally impossible to quantify risk because of:-

- Problems of measurement
- Failure to take account of risk compensation, (people compensate for greater safety by taking more risks.)

Problems of measurement - A genius' s view of risk

"If a guy tells me that the probability of failure is 1 in 10^5, I know he's full of crap."

Richard P. Feynmann, Nobel Laureate commenting on the NASA Challenger disaster.

Risk compensation

Problem:-

- 500 motorcyclists a year are killed in accidents in the U.K.

Solution

- Ban motorcycles

Discuss ...

The risk thermostat, (J. Adams)

This view of risk argues:-

- Everybody has a propensity to take risk
- This propensity varies between people
- Risk-taking is influenced by the rewards
- Perceptions of risk are influenced by experience of losses - one's own and others
- Risk-taking involved a balancing between the propensity to take risk and the perceived risk

The dance of the ‘ risk thermostats’

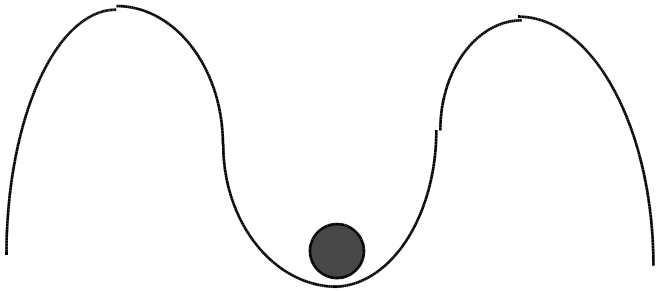
Interaction in society involves:-

- Continuous dance of every individual’ s risk thermostat and interaction with other risk thermostats
- Underlying chaos which further undermines quantification

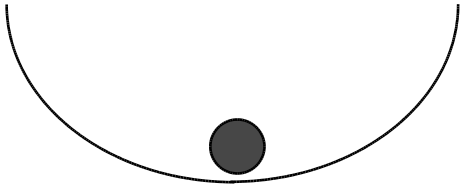
If as it seems quantification seems impossible, are there any useful patterns ?

The 4 managerial views of nature, (Holling)

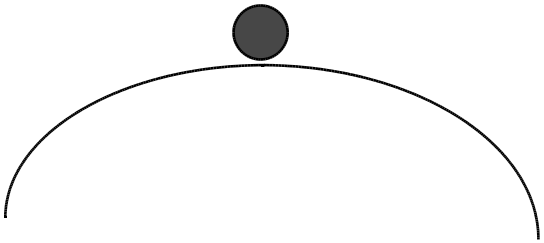
Nature capricious,
fatalist



Nature perverse /tolerant,
interventionist



Nature benign,
laissez-faire



Nature ephemeral,
precautionary

Different rationalities

- Rational argument is based upon logic, mathematics and grammar
- In an uncertain world, rational arguments are constructed on premises beyond rationality
- People apply different views of nature in a rational argument

Different rationalities

The four basic rationalities are:-

- Individualist
  - relatively free from control by others and seek to control their environment. Example - hacker.
- Hierarchist
  - inhabit a world of strong group boundaries and hierarchical structures. Example - quality manager
- Egalitarian
  - Strong group loyalties but little respect for externally imposed rules. Example - users
- Fatalist
  - Resigned to their fate and make no effort to change it. Example - trombone player.

Different rationalities

If asked how we manage risk, the reactions of the four basic rationalities are:-

- Individualist
  - asserts we are already over-regulated and we should leave it to market forces
- Hierarchist
  - says we need more research but things are basically OK
- Egalitarian
  - urge precaution and press for urgent action
- Fatalist
  - watch television and buy lottery tickets

Conclusions about risk

- It is almost impossible to quantify risk or at least we have totally failed to achieve it so far
- Realising that each person approaches a risk with some dynamic mixture of the four basic rationalities is important to understanding the inherently associative nature of risk.

Conclusions about risk, (J. Adams)

- Everyone else is seeking to manage risk too
- Everybody is guessing. If they knew, its not risk
- Guesses are extremely influenced by beliefs
- The behaviour of others and the behaviour of nature are your risk environment
- Unless people's propensity to take risk is reduced:-
  - Safety intervention simply leads to responses which re-establish the level of risk
  - Safety intervention redistributes risk but does not reduce it
- Science will continue to invent new risks
- In the dance of the risk thermostats, the music never stops

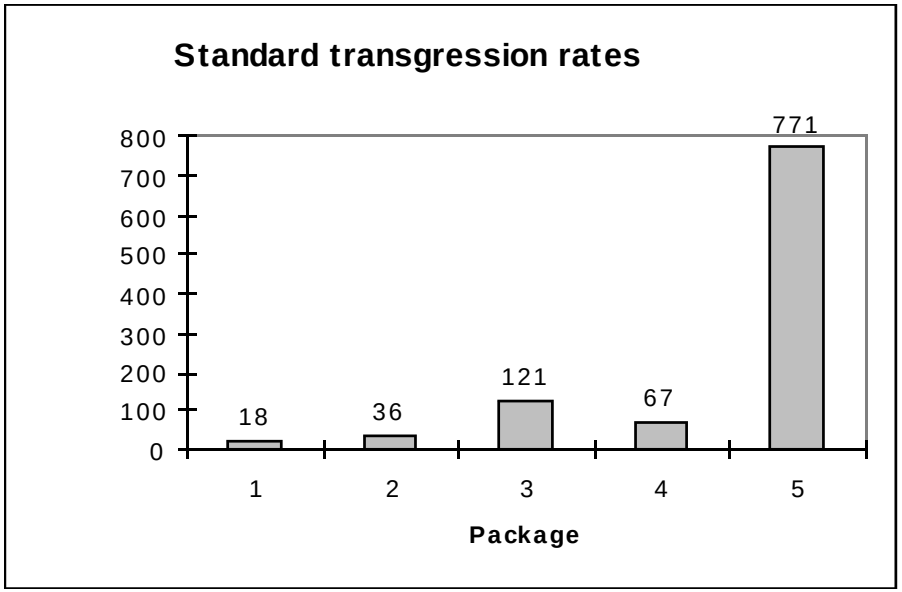
Relevance

So what does all this have to do with risk and benefit in modern software engineering and how does it contribute to testing economics ?

Overview

- **An introduction to Risk**
- **Examples and sources of risk and failure**
- **Software Risk Mitigation**

Hierarchists v. Individualists



Trangression rates of standards in executable lines per transgression, Fortran77 and C commercial systems, Hatton (1995)

Some personal web observations:-

- A significant percentage of web sites have basic Javascript errors
- Internet financial fraud is thought to be much higher than normal financial fraud
- The author's bank allowed the user to continue in the clear without security as an 'option' in its first release.
- Many web-sites exhibit unusual behaviour

Web failures - short sample from 14/09/00-5/10/00

- **ID Error halts Egg's online share dealing**
  - The site allowed logon with wrong-IDs
- **Netcetera web hosting problem**
  - This allowed companies confidential files to be viewed by other companies
- **E-mail bugs on BTs Talk21 line**
  - These allowed access to other users in-boxes

An example of an automobile system failure:-

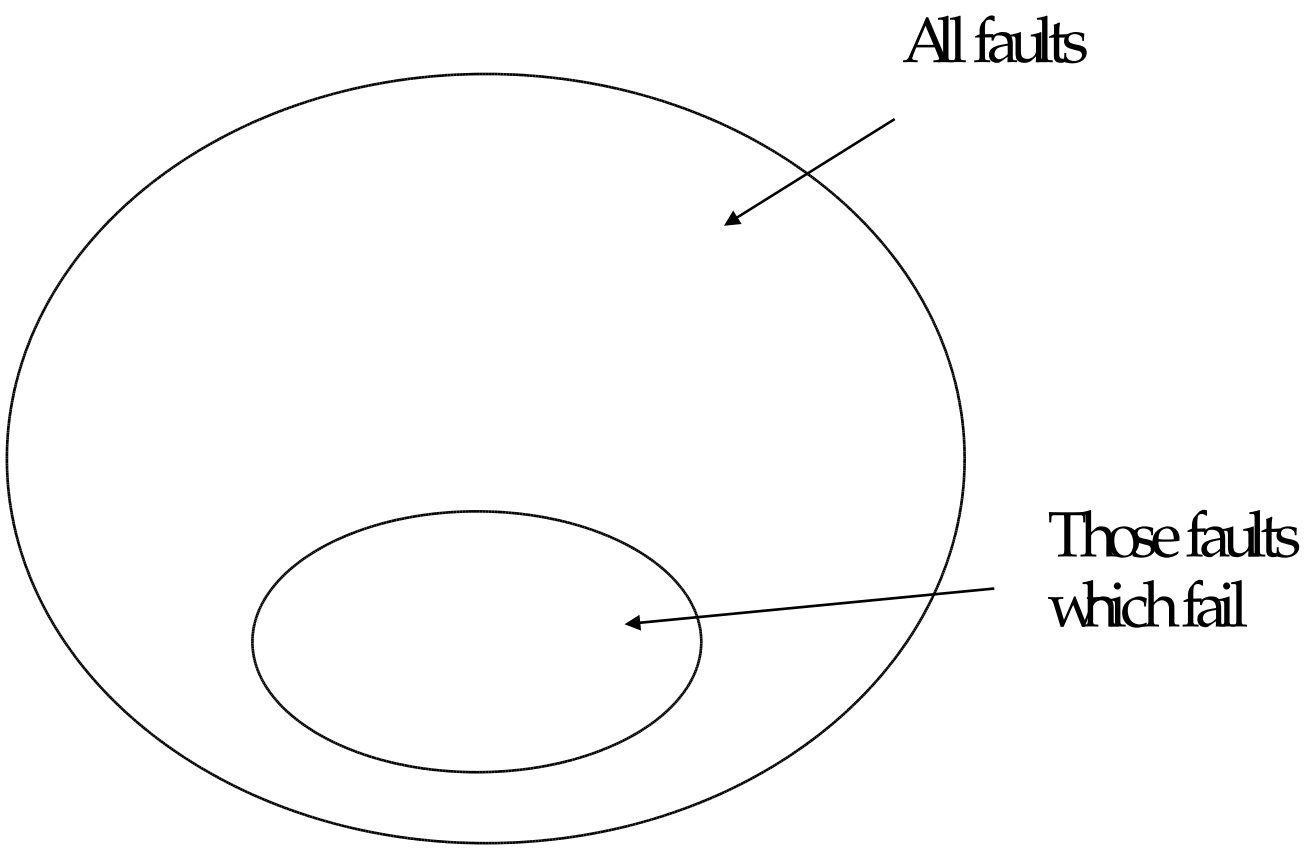
- 22/July/1999. General Motors has to recall 3.5 million vehicles because of a software defect. Stopping distances were extended by 15-20 metres.
- Federal investigators received almost 11,000 complaints as well reports of 2,111 crashes and 293 injuries.
- Recall costs ? (An exercise for the reader).

Software safety defect hits Ford

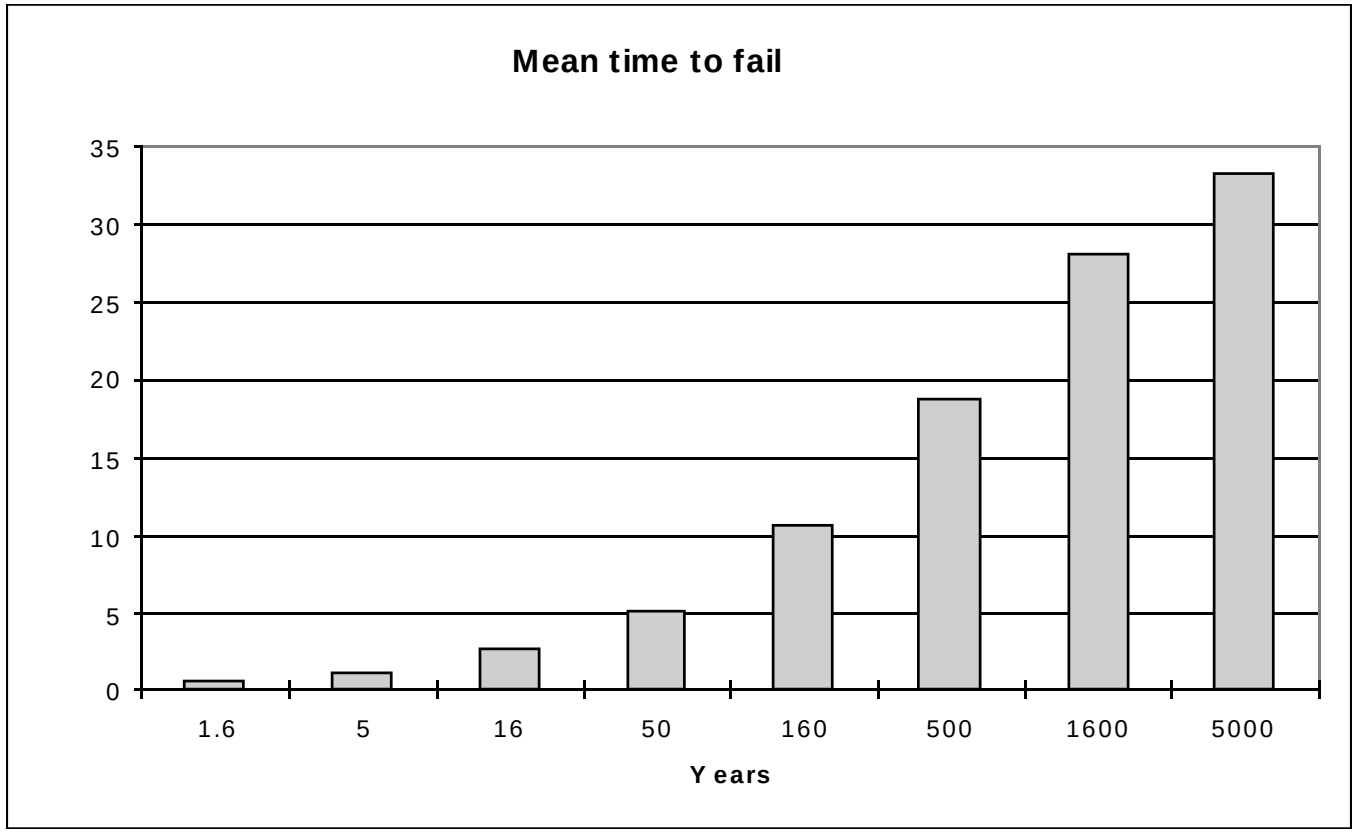
- **14/Sep/2000**. Production of year 2001 models of Ford Windstar, Crown Victoria, Mercury Grand and Lincoln stopped because of software defect causing airbags to deploy on their own and seatbelts to tighten suddenly.
- This stopped production for several days at Ford of Canada and other sites. At least 15,000 cars have to be recalled and fixed.
- The software was out-sourced.
- Recall costs are not yet known.

What can we find in common between web software and embedded control systems?

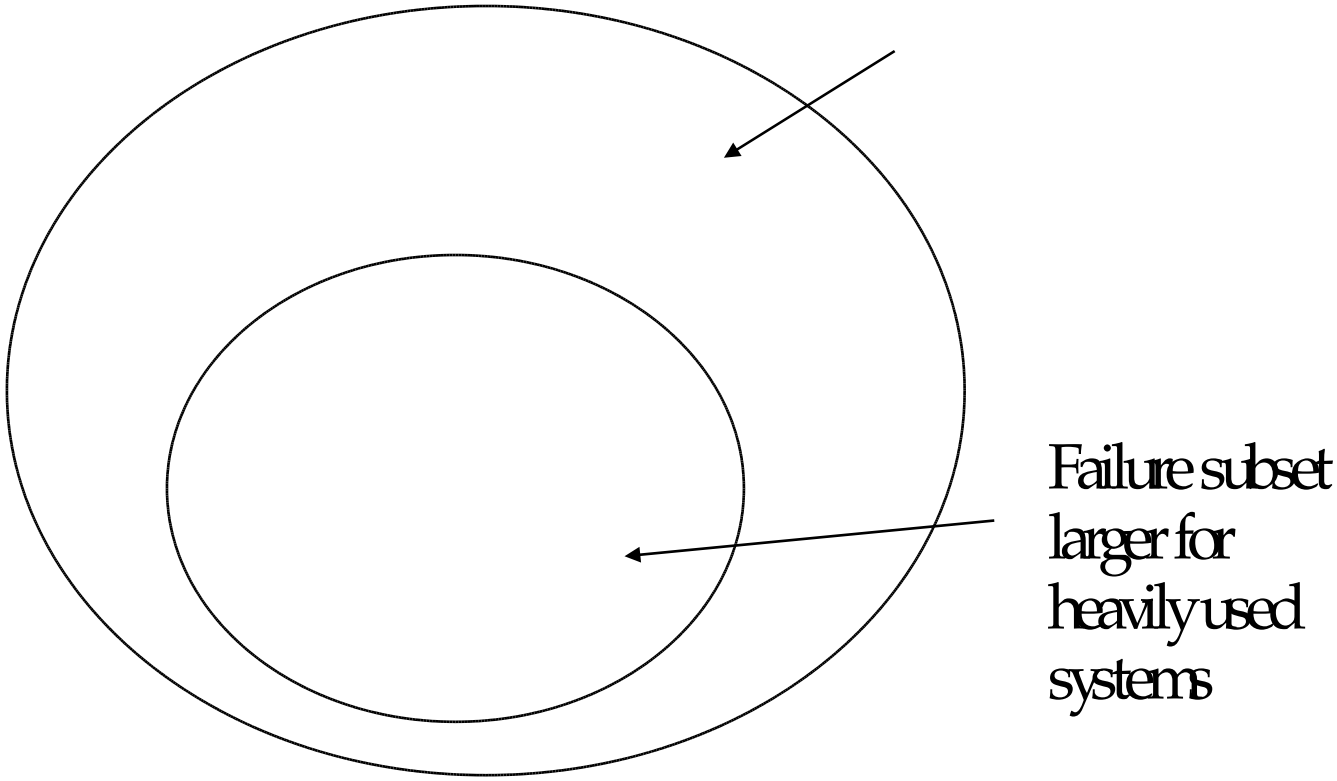
Where and how do defects occur historically ?



Mean time to fail in Adams (1984)



Where and how do defects occur historically ?



Mean time to fail in Adams (1984)

- **This study found that:-**
  - ~33% of all faults only failed < once every 5000 execution years
  - The most common failures, (> once every 5 years) were caused by only 2% of the faults.
  - Any correction had about a 15% chance of introducing a problem at least as big into the system.

Some notes

Time to failure:-

- In an air-traffic control system with 10 copies running 7x24x365, the first 5000 year failures would take 500 years to appear
- In an embedded control system in a car with say 1,000,000 copies around the world, they will first appear in about 4 days.

Web sites which are heavily used exhibit exactly the same kind of behaviour. (Note also that these are exactly the circumstances favouring inspections.)

An example of the Ed Adams' injection effect

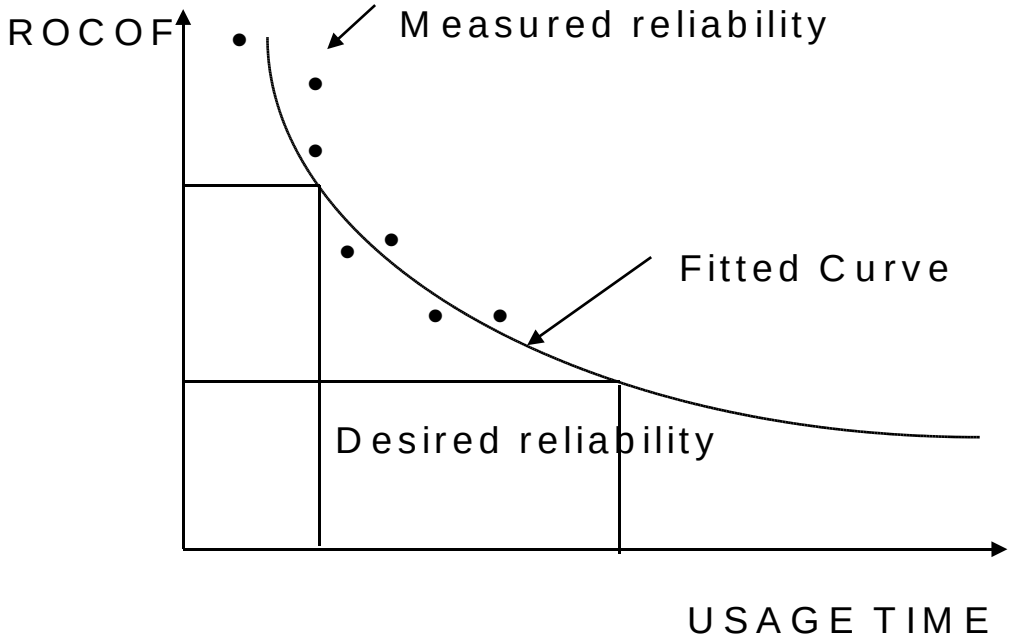
In October, the UK National Air Traffic Control authority admitted that the number of problems in its new centre at Swanwick in Hampshire followed this pattern:-

- 5/2000, 550
- 8/2000, 200
- 9/2000, 217

(This represents a re-injection rate of roughly 6% which is quite good. No allowance appears to have been made for this effect.)

Risk and Benefit

How do you test a system intended for very heavy use ?

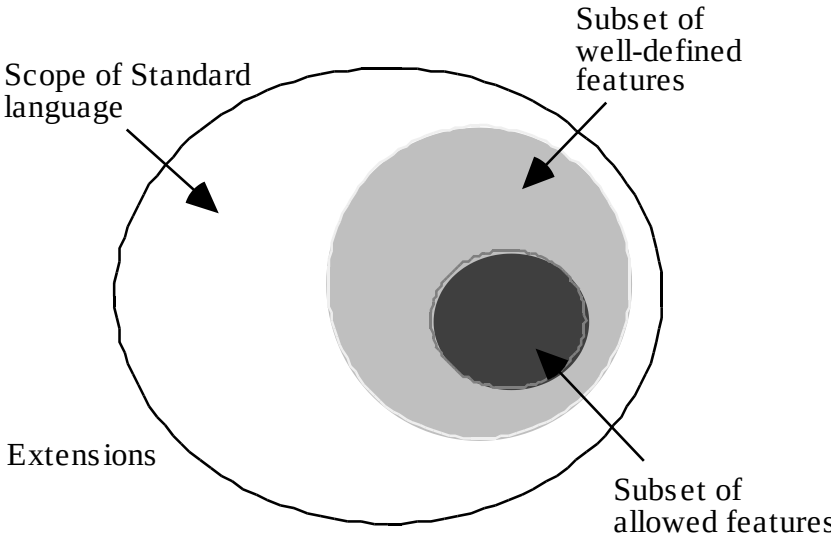


Relevance to risk discussion

There is clear evidence with both embedded control systems and web development that the increased risks produced by unusually large usage are being ignored.

In essence, everybody temporarily becomes a fatalist.

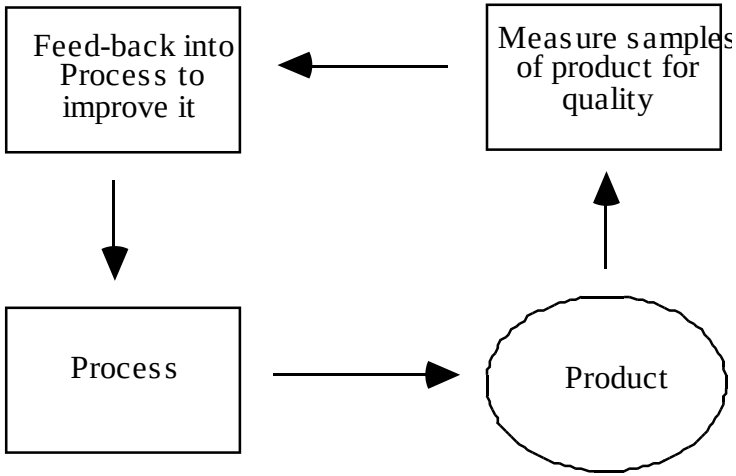
The need for subsetting programming languages



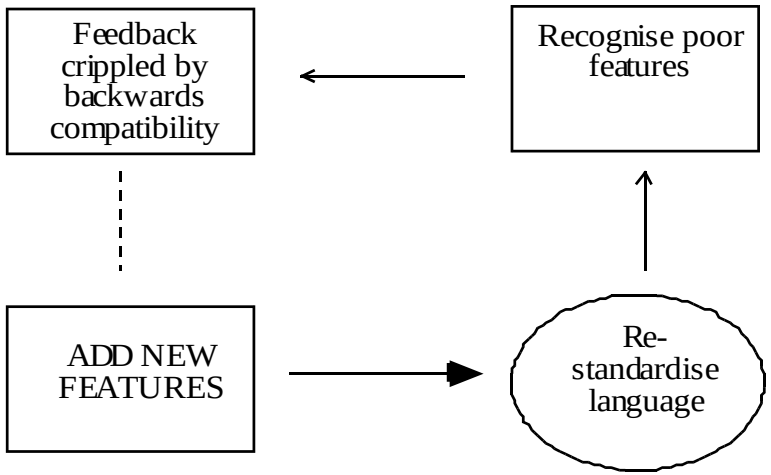
Do languages improve with time ?

- **Things get worse with time. The following areas of C are problematic because the committee could not agree:**
  - At standardisation in 1990 (197 items)
  - At re-standardisation in 1999 (366 items)
- **By comparison, C++99 contains the words:-**
  - Undefined, 1825 times
  - Unspecified, 1259 times.

Control Process feedback - an example of a hierarchist technique



Why languages can't improve



Programming languages are designed by **individualists**.

Control process feedback is a tool used by **hierarchists**.

Language standardisation ...

- **Language standardisation disobeys control process feedback in several important ways:-**
  - It is characterised by often unconstrained creativity
  - It completely ignores measurement
  - The ‘ must not break old code’ rule means feedback is crippled so although things are continually added, little gets taken out in practice.

The problem of course is that we are trying to use a hierarchist technique on an individualist technology.

Overview

- **An introduction to Risk**
- **Examples and sources of risk and failure**
- **Software Risk Mitigation**

What can we do ?

As we saw earlier in the discussion of risk,

- We must reduce the propensity of software managers and engineers to take risks
  - By making managers more aware of the cost of failure
  - By making engineers and managers more aware of the ability of testing technology to reduce the cost of failure
- Will it help to produce more reliable software ?
  - Probably not. Every observation of society suggests that risk compensation usually balances risk mitigation. In software engineering, an improvement in basic reliability will probably be offset by the addition of new features

A prediction

Improvements in software testing will not in general lead to improved reliability. They will simply lead to more features at least in the foreseeable future.

If we judge such a system to be better then we are making progress, however if we are building critical systems, feature introduction must take second place to reliability improvement.

Both feature introduction *and* reliability improvement do not seem to be an option.