UKSIM' 99, 7-9 April, 1999

"The influence of software defects on simulations"

by

Les Hatton

Oakwood Computing, U.K.

and Computing Laboratory, University of Kent

lesh@oakcomp.co.uk

Last update: 7th April 1999

An overview

- **Overview**
- **The T experiments**
- **T1: static study results**
- **T2: dynamic study results**

A wider view of software failure

1980' s :
30-5-1 defect density rule independent of language.

1976-1990:
NASA Goddard defect density almost constant with time. but spread getting smaller.

1995:
Airbus A340 is first commercial aircraft to exhibit significant inflight software failure.

1995:
Software in consumer electronics *doubling* every 18 months

2000-:
Erk !

1984:
Ed Adam' s work at IBM - some faults never fail and correction leads to more failures.

1983:
NASA show 50% of all maintenance is corrective, (Arnold).

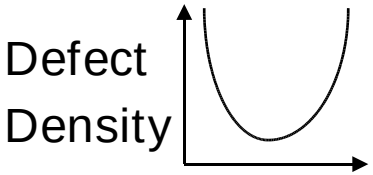
1996:
Ariane 5 sets back European Monetary Union somewhat.

1990-1999:
Cars, radios, lifts, entertainment systems and so on all begin to exhibit significant software failure

A personal view of software failure

1995:
Around 40% of
all dynamic failure
is statically detectable.

1989-1998



1990-95:
T1: ~10 static faults/KLOC
in F77, C. (C++ bit worse)

1990-1993:
T2: 9-version dynamic experiment.
Only 1 sig. fig. agreement left at end.

1995-1998:
Win' 95 1 defect every 42 mins.
Mac - 1 defect every 188 mins.
Unix. Linux - Almost never.

1986-1988:
Porting same F77 package
gave 4 sig.fig. agreement
on different platforms.

1997:
Formal methods => 3:1 better
Static fault highly correlated
to dynamic failure.

1996-8:
O-O/C++ has comparable defect
density but 2-3 times corrective
maintenance cost.

1995-1999:
Code inspections
repeatedly implicated in high-
quality systems.

An overview

- **Overview**
- **The T experiments**
- **T1: static study results**
- **T2: dynamic study results**

The T-experiments

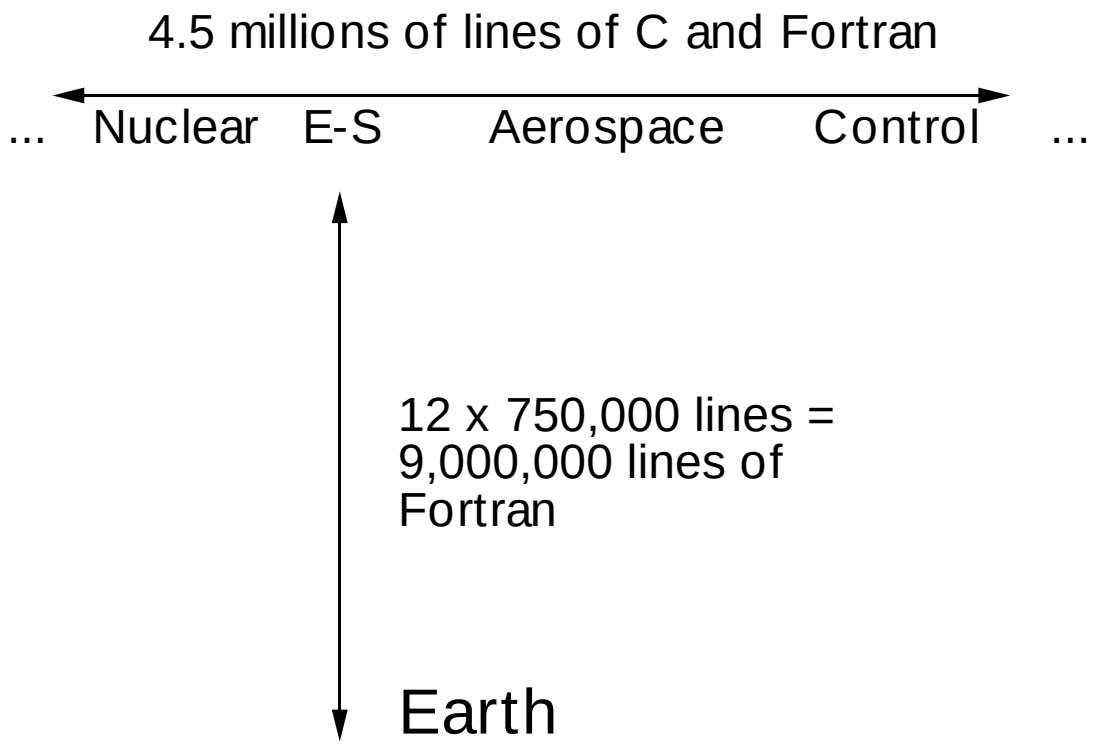
Multi-industry study using static inspection, 1990-1992

... Nuclear E-S Aerospace Control ...

Single-industry study using N-version techniques, 1990-1993

Earth Science

The T- experiments



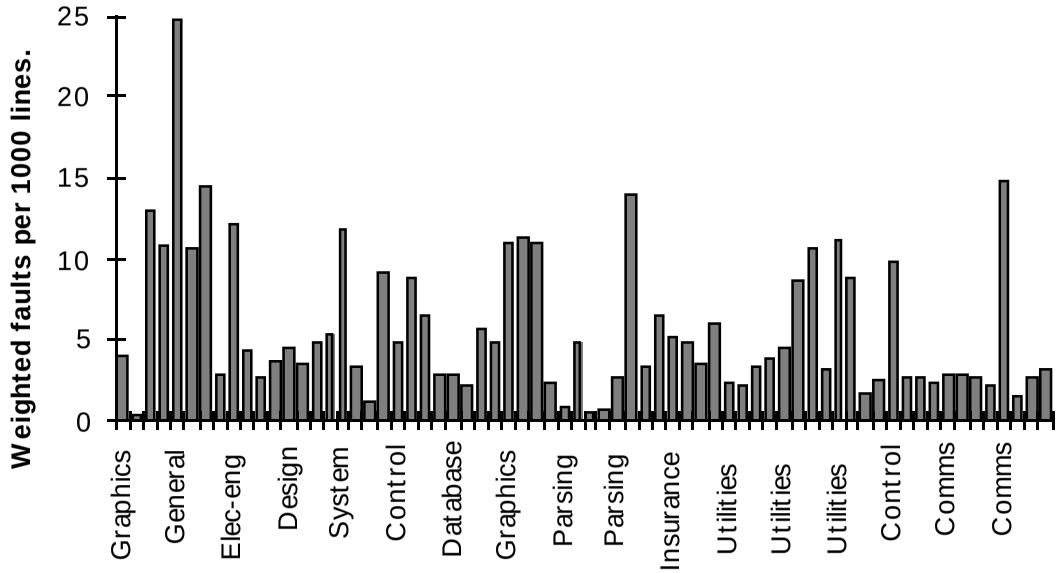
An overview

- **Overview**
- **The T experiments**
- **T1: static study results**
- **T2: dynamic study results**

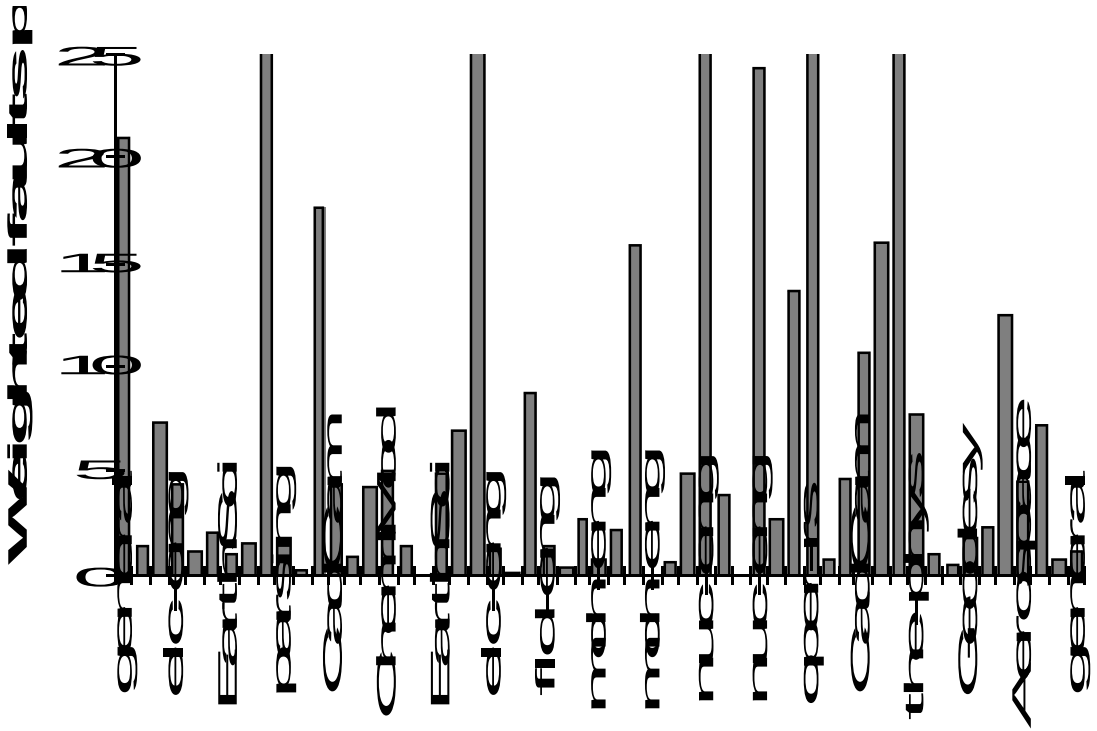
Static testing - what are we looking for ?

- **In programming languages, poorly defined behaviour comes from:**
  - Formally undefined behaviour (e.g. 197 issues defined in ISO C90).
  - Accidentally undefined behaviour, (1 issue in Pascal, ~119 issues in C, ~80 in extended Pascal, ~1400 in Ada)
  - Well defined behaviour which people make a mess of anyway, (e.g. Jan 1990 AT&T fiasco).
- **All languages are stricken to a *quantifiable* extent with these problems.**

Avoidable (i.e. detectable) faults in C applications



Avoidable (i.e. detectable) faults in F77 applications



An overview

- **Overview**
- **The T experiments**
- **T1: static study results**
- **T2: dynamic study results**

T2 details

- 9 independently developed commercial versions of same ~750,000 F77 package of signal processing algorithms.
- Same input data tapes.
- Same processing parameters, (46 page monitored specification document).
- All algorithms published with precise specification, (e.g. FFT, deconvolution, finite-difference wave-equation solutions, tridiagonal matrix inversions and so on).
- All companies had detailed QA and testing procedures.

Basic goals of experiment

- **Overall goals were:**
  - To estimate the magnitude of disagreement.
  - To see what form disagreement took.
  - To identify poorly implemented processes.
  - To attempt to improve agreement by feedback confirming nature of fault.
  - To preserve complete confidentiality.

Data analysis

- **Analysis goals were:**
  - Analyse at 14 "primary" calibration points and 20 "secondary" calibration points.
  - Analyse data in multiple windows.
  - Use two sets of independently developed analysis software to improve confidence.

T2 Results

- **The accompanying slides illustrate:**
  - Only 1 significant figure agreement after processing.
  - Disagreement is non-random.
  - Feedback of anomalies along with other evidence confirms source of disagreement as software failure.

T2 overall perspective

Seismic processing software environment	Number of significant figures agreement
32 bit floating point arithmetic.	6
Same software on different platforms, same data.	4
Same software on same platform, 5-1 lossy compression.	3-4
Same software subjected to continual 'enhancement'	1-2
T2: different software, same specs, same data, same language, same parameters.	1

Conclusions

- **Ample evidence that our expectations of software accuracy are far too high.**
- **Ample evidence that software failure is the problem.**
- **Ample evidence that no great paradigm shift has helped much, e.g. O-O, formal methods.**
- **Ample evidence that choice of language doesn't make much difference *but fluency does*.**
- **Ample evidence that much software failure is avoidable by techniques *we already know how to do but for some reason do not use*.**