"Why is Linux so reliable?"

ASM 2000, San Jose, 8th Mar., 2000

by

Les Hatton

Oakwood Computing, Surrey, U.K.
and
Computing Laboratory, University of Kent, UK

lesh@oakcomp.co.uk

Version 1.1: 02/Mar/2000

Overview of talk

- **What is Linux?**
- **How reliable is it?**
- **Why ask the question?**
- **How did it get this good?**

A short history of Linux

- Started in 1991 by Linus Torvalds who wanted a cheap reliable OS for his PC
- It has now grown to be supported by hundreds of very able developers working voluntarily on the Internet
- It is freely distributed over the Internet and also packaged by a number of organisations, Red-Hat, SUSE, Mandrake, Caldera, Debian and Slackware.

A short history of Linux

- It is a full multi-tasking, multi-threaded symmetric multi-processing operating system
- It is quoted as being responsible for running around 54% of the servers on the Web using Apache
- It is freely distributed under the GNU Public Licence
- Its reliability is legendary, (more later)

A short history of Linux

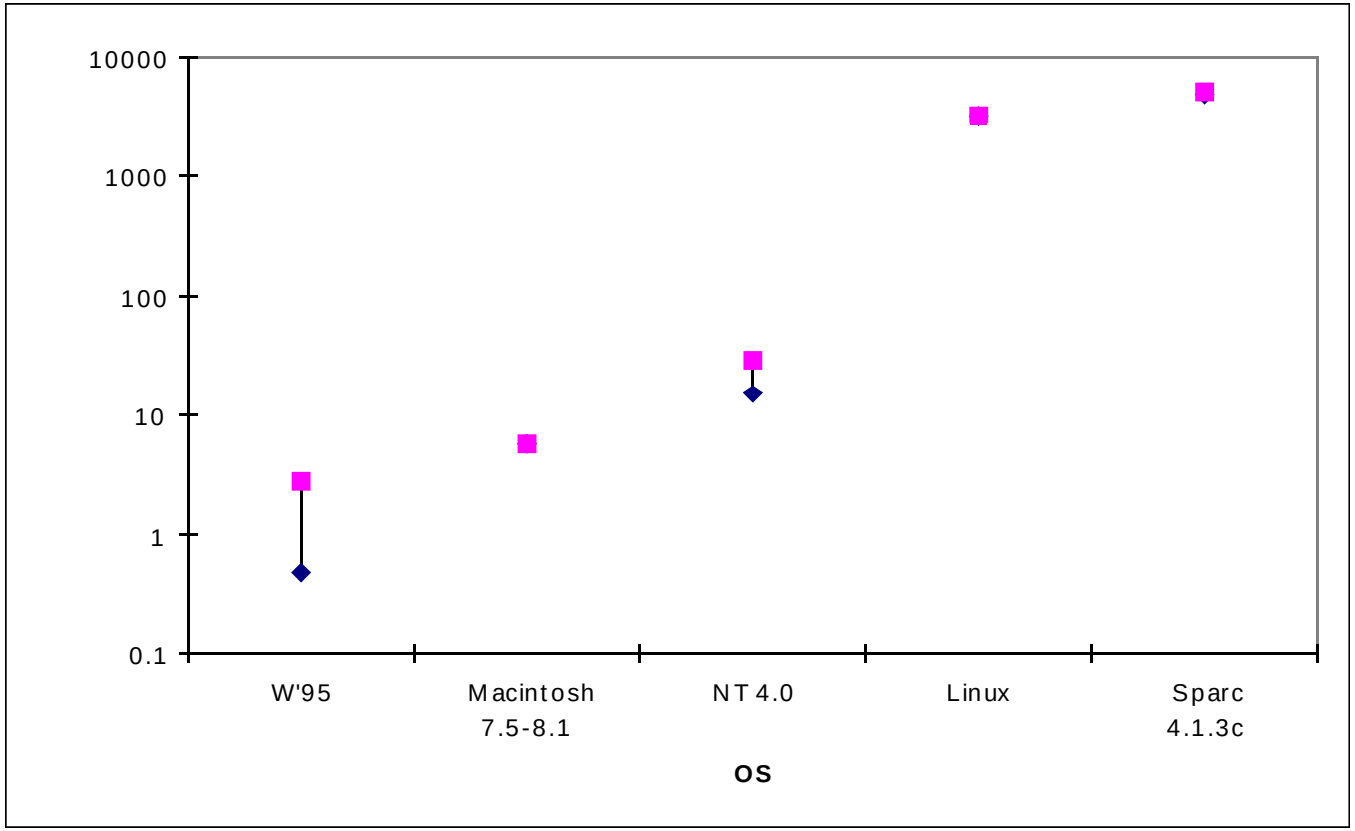
- It should really be known as GNU/Linux. (The GNU organisation was founded by Richard Stallman in the '80s).
- It is very compact and will run comfortably in 16 Mb on a 386 PC
- In the last year, Oracle, IBM and many other major industry companies have begun to support products on it.

- It is very highly portable
- It is fully Posix compliant
- The number of applications is growing at an amazing rate
- It distinguishes between 'development' and 'stability' releases

Overview of talk

- **What is Linux?**
- **How reliable is it?**
- **Why ask the question?**
- **How did it get this good?**

Linux reliability in MTBF



Mean Time Between Failures of various operating systems

Personal data due to Hatton (2000)

Other sources of data:-

- Central Virginia power distribution, (Petree (1999))
  - " The [Linux] systems software has never failed nor caused us a single problem through thousands of hours of continuous uptime."
- Banking, (Shoham (1998))
  - " During stress testing, we discovered 2 separate OS bugs in MVS (IBM's venerable mainframe operating system), but no problems and no bugs in the Linux servers."
  - " It is interesting to note that the least expensive component of the system and the one with no formal vendor support has been the source of the fewest (i.e. zero) problems."

Other sources of data:-

- Internet server, (Ogbuji (1998))
  - " For several months a single Pentium Pro-based server running Linux ran mail, DNS, central logging, IMAP, SMB and WWW for over 1000 users with little or no downtime."
- Driving a car round the Mille Miglia, Italy, (Bertozzi (1999))
  - " The number of faults found in the (Linux) system components was zero over the last two years."
  - Note that this electric car achieved over 90% automatic (hands-off) driving using two CCD cameras and electronic controls all handled by a Linux PC in the car.

Applications:-

- Open source applications
  - These tend to be very reliable. No defects so far found in software development after 3300 hours. Star Office also appears to be excellent.
- Applications ported from PC
  - Corel WordPerfect 8, (Linux edition) immediately core-dumped when trying to open the same 1.4 Mb. Word file on which Star Office performed effortlessly. So, no change there.

Note that the reliability does not seem to be at the expense of performance:-

- Interface tests of commercial product. (20 rapid button presses to launch the application):-

Time taken to return (Windows NT 4.0)	Time taken to return (Linux Red-Hat 6.0)
4.0 seconds	0.4 seconds

Overview of talk

- **What is Linux?**
- **How reliable is it?**
- **Why ask the question?**
- **How did it get this good?**

Why ask the question ?

- **Process quality v. product quality**
- **The CMM and Linux**
- **What is unusual about Linux development ?**

Process quality v. product quality

- There has been a belief for a number of years that good process quality is a necessary condition for good product quality. (In some quarters it is considered sufficient also)
- Good process quality is built around process models such as the CMM, ISO 9001 and so on

Why ask the question ?

- **Process quality v. product quality**
- **The CMM and Linux**
- **What is unusual about Linux development ?**

The CMM and Linux

- The CMM is a five-level process model:-
  - Level 1: Chaotic
  - Level 2: Repeatable
  - Level 3: Defined
  - Level 4: Managed
  - Level 5: Optimised

In an assessment of Linux, Hatton (1999) concluded that Linux has key failures at all levels including level 1.

- Example CMM Level 1 failures
  - Requirements management**
  - Software project planning*
  - Software project tracking and oversight*
  - Software subcontract management
  - Software quality assurance

* Not so relevant as there are no 'official' deadlines

** Initially not relevant as it was implementing a known API

- Example CMM Level 2 failures
  - Organisational process focus
  - Organisational process definition
  - Integrated Software Management

These are essentially management issues. (Linux isn't strong on management).

Why ask the question ?

- **Process quality v. product quality**
- **The CMM and Linux**
- **What is unusual about Linux development ?**

What is unusual about Linux development ?

- One or two technologies are exemplary
  - Configuration and build management is arguably amongst the best in the world. It is distributed and uses core open source technologies such as RCS, CVS and RPM.
  - The open source model leads to very large scale code inspection
- Communication is very good, (they are volunteers)
- They naturally work in small groups (up to around 6)
- The quality of the developers tends to be outstanding

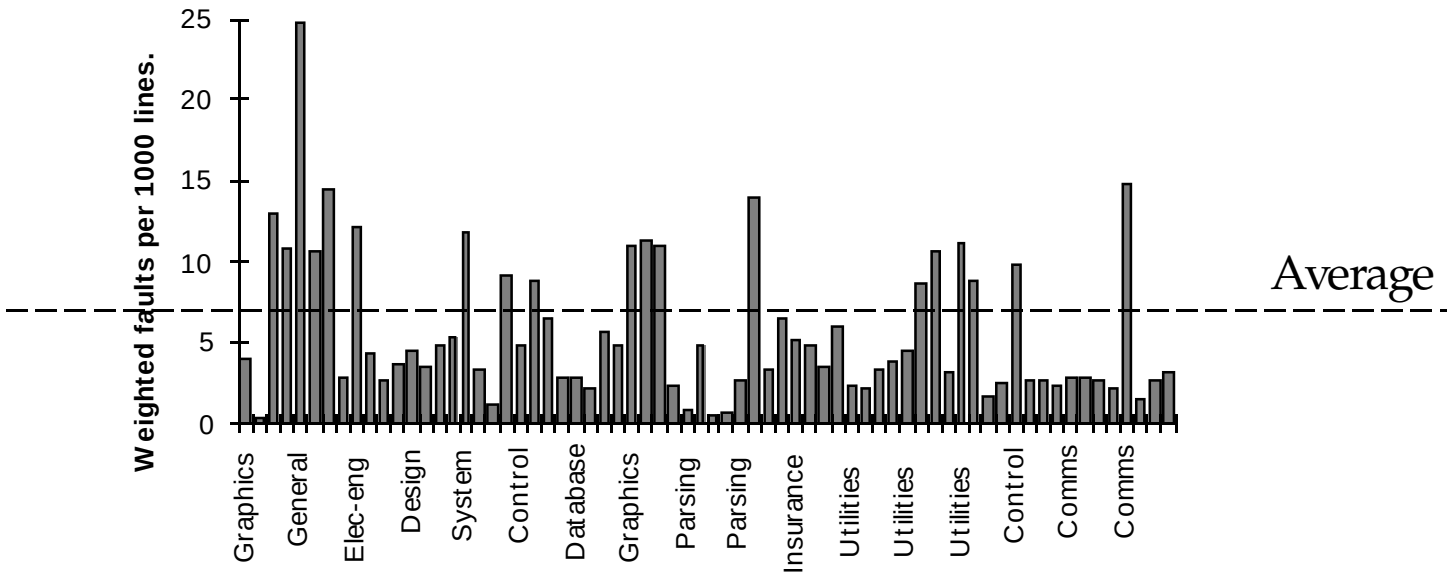
The importance of high-quality developers

This has been noted on many occasions

- Brooks (1975) first talked about the importance of engineer quality.
- Knight and Leveson (1986). Independent versions of the same program requirements in Pascal varied in size by more than a factor of 3 and dramatically in reliability
- Curtis et. al (1988) talked about the importance of central architects in maintaining coherence
- Coplien (1994) emphasised this in the development of Borland's Paradox
- Hatton (1995) noted dramatic variations in injected defect density by different programmers implementing the same requirements in parallel in the same programming language

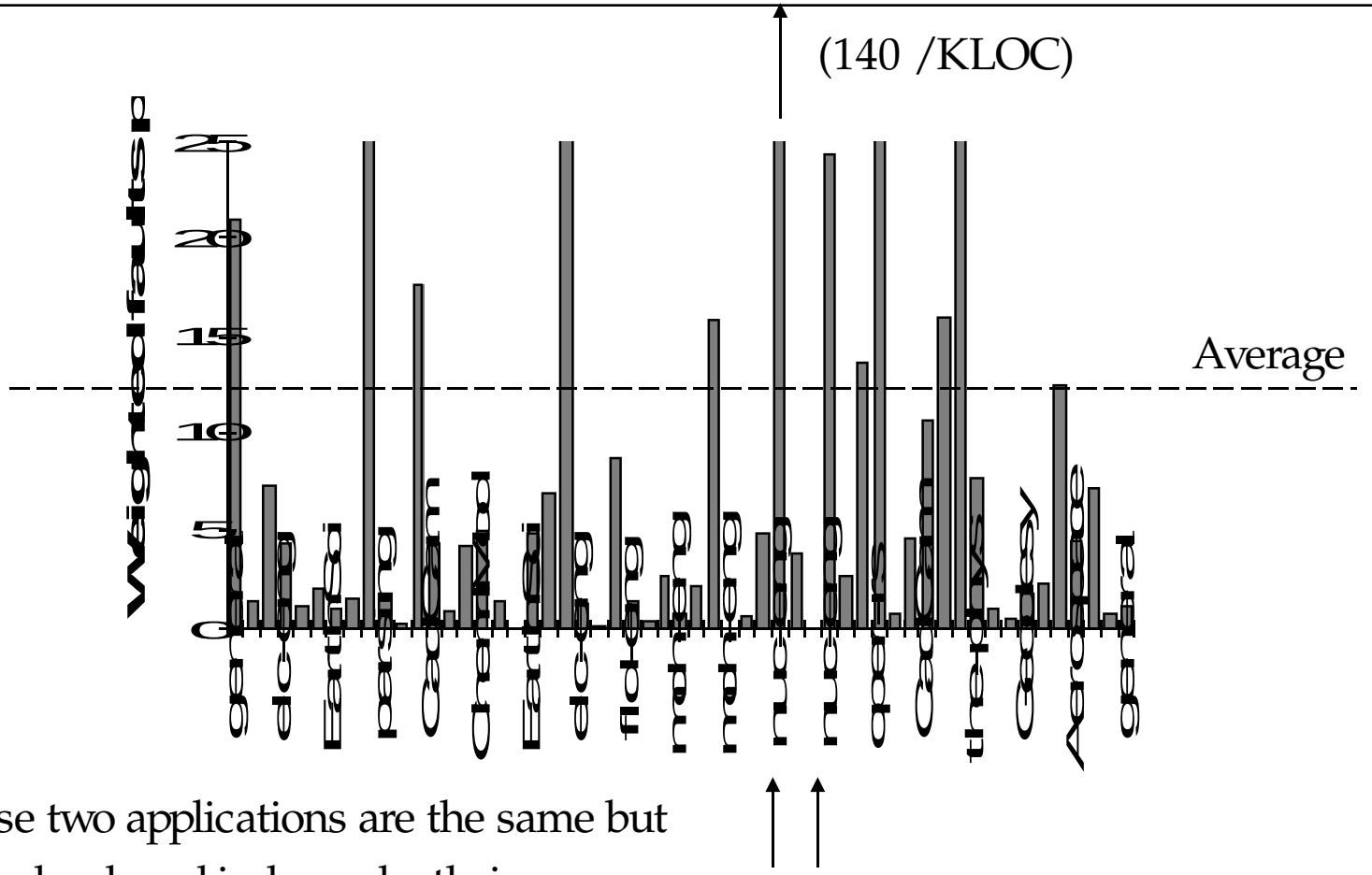
- Prechelt (1999) noted in Java / C++ / C efficiency comparisons that the programmer made far more difference than the language.

Static defect rates in C applications



There is no obvious relation between these and whether or not the company concernedused ISO 9001

Occurrence rates in F77 applications



These two applications are the same but were developed independently in different companies

Overview of talk

- **What is Linux?**
- **How reliable is it?**
- **Why ask the question?**
- **How did it get this good?**

How did it get this good ?

Note the following:-

- Relevance to commercial organisations ?
  - The Linux model has no deadlines, only good intentions
  - It operates strictly on the open source model (OSS) which most intellectual property lawyers are genetically incapable of understanding
- It appears to rely on a small number of excellent technologies, all informally supplied:-
  - Individual engineer excellence and coherence
  - State-of-the-art configuration control
  - Large-scale code inspections
  - Commitment

An important addendum on Apache

Note the following:-

- In parallel with this work, Mockus, Fielding and Herbsleb (2000) at Lucent studied Apache. They concluded:-
  - There are around 400 developers but the top 15 developers contributed 83% of the changes, 88% of added lines and 91% of deleted lines, (c.f. " The Mythical Man Month, Brooks, (1975)).
  - The released defect density was a little worse than comparable high-reliability commercial products but the initial defect density was much lower confirming the benefit of inspections and balancing the much lower level of system testing.
  - Code ownership is based around recognition of expertise.
  - Problems are turned round very rapidly, (> 50% within 1 day).
  - Corrective maintenance was around 11% of all maintenance.

An important addendum on Apache ...

Lessons to learn:-

- As in this work, Mockus, Fielding and Herbsleb at Lucent concluded:-
  - A core of experienced domain-aware developers is fundamental to reliability
  - Large scale code inspections even informally carried out are extremely effective at reducing the defect injection rate before dynamic testing even begins.

An important addendum on Apache ...

Hypotheses to test further:-

- Mockus, Fielding and Herbsleb at Lucent also hypothesised:-
  - In successful OSS development, a group larger by an order of magnitude than the core will repair defects and a yet larger group (by another order of magnitude) wil report problems.
  - Even with a strong core, if the surrounding maintenance cloud is missing, the development will die. (This is the Darwinian effect described by Hatton (2000)).
  - Defect density in OSS development will generally be lower than commercial code which has received a comparable level of dynamic testing.
  - OSS developments exhibit very rapid responses to customer problems.

Conclusions

Note the following:-

- It is possible to produce highly reliable software :-)
- The Linux model once again emphasises the importance of the quality of individual engineers
- Other facets of the model may not be so relevant to commercial developers
- It will be interesting to see to what extent we can take Linux reliability. In the critical applications of the 21st century, very high reliability will be necessary. This may be the only way of achieving it.

Collective source, collective responsibility