The Good, the Bad, and the Ugly?

Structure and Trends of Open Unix Kernels

Dr. Wolfgang Mauerer, Siemens AG, CT SE 2
Corporate Competence Centre Embedded Linux
wolfgang.mauerer@siemens.com

Overview

SIEMENS

Outline

Introduction

Observations

- Users won’t really notice different kernel underneath their systems (spark plugs!)
- Developers will mostly neither.

Questions

- Why so many different kernels?
- How do they differ?
- *Quantitative* assessment possible?
- Which one’s the best kernel?

Introduction

Observations

- Users won’t really notice different kernel underneath their systems (spark plugs!)
- Developers (*except system and spark plug developers*) will mostly neither.

Questions

- Why so many different kernels?
- How do they differ?
- *Quantitative* assessment possible?
- Which one’s the best kernel?

Introduction

Observations

- Users won't really notice different kernel underneath their systems (spark plugs!)
- Developers (*except system and spark plug developers*) will mostly neither.

Questions

- Why so many different kernels?
- How do they differ?
- *Quantitative* assessment possible?
- Which one's the best kernel?

Introduction

Observations

- Users won’t really notice different kernel underneath their systems (spark plugs!)
- Developers (*except system and spark plug developers*) will mostly neither.

Questions

- Why so many different kernels?
- How do they differ?
- *Quantitative* assessment possible?
- ~~Which one’s the best kernel?~~

SIEMENS

Outline

Evaluated Systems: Linux and OpenSolaris

- Started: 1991
- License: GPL
- Version Control: GIT

- Started: 2008 (1991)
- License: CDDL
- Version Control: Mercurial

The BSD “family”

- Started: 1993
- License: BSD
- Version Control: Subversion

- Started: 1991
- License: BSD
- Version Control: CVS

- Started: 1996 (1991)
- License: BSD
- Version Control: CVS

Outline

Unbiased?

No flame wars intended. . .

Feature Comparison

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

Feature Comparison

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

Feature Comparison

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

SIEMENS

Feature Comparison

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

Feature Comparison

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

Feature Comparison

SIEMENS

Commonplaces

- Yes, OpenSolaris focusses on tracing, zones, and ZFS
- Yes, Linux still focusses on . . . everything
- Yes, NetBSD is really about multi-arch support
- Yes, OpenBSD is about security and paranoia
- Yes, FreeBSD would like Linux's device drivers

Alternatives

- Performance testing
- Tick list on Wikipedia

Outline

Quantitative measurements

SIEMENS

Quantitative != Useful

- Formal complexity vs. real complexity
- Line counting == bean counting?
- Reliability of direct comparisons?

Question

Can measurements provide *meaningful* insights?

A Tale of Four Kernels

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Patision 76, GR-104 34 Athens, Greece
dds@aueb.gr

ABSTRACT

The FreeBSD, GNU/Linux, Solaris, and Windows operating systems have kernels that provide comparable facilities. Interestingly, their code bases share almost no common parts, while their development processes vary dramatically. We analyze the source code of the four systems by collecting metrics in the areas of file organization, code structure, code style, the use of the C preprocessor, and data organization. The aggregate results indicate that across various areas and many different metrics, four systems developed using wildly different processes score comparably. This allows us to posit that the structure and internal quality attributes of a working, non-trivial software artifact will represent first and foremost the engineering requirements of its construction, with the influence of process being marginal, if any.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Software process models*; D.2.8 [**Software Engineering**]: Metrics—*Product metrics*

General Terms

Measurement

1. INTRODUCTION

Quantitative measurements

Quantitative != Useful

- Formal complexity vs. real complexity
- Line counting == bean counting?
- Reliability of direct comparisons?

Question

Can measurements provide *meaningful* insights?

A Tale of Four Kernels

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
Patision 76, GR-104 34 Athens, Greece
dds@aueb.gr

ABSTRACT

The FreeBSD, GNU/Linux, Solaris, and Windows operating systems have kernels that provide comparable facilities. Interestingly, their code bases share almost no common parts, while their development processes vary dramatically. We analyze the source code of the four systems by collecting metrics in the areas of file organization, code structure, code style, the use of the C preprocessor, and data organization. The aggregate results indicate that across various areas and many different metrics, four systems developed using wildly different processes score comparably. This allows us to posit that the structure and internal quality attributes of a working, non-trivial software artifact will represent first and foremost the engineering requirements of its construction, with the influence of process being marginal, if any.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Software process models*; D.2.8 [**Software Engineering**]: Metrics—*Prod*

General Terms

Measurement

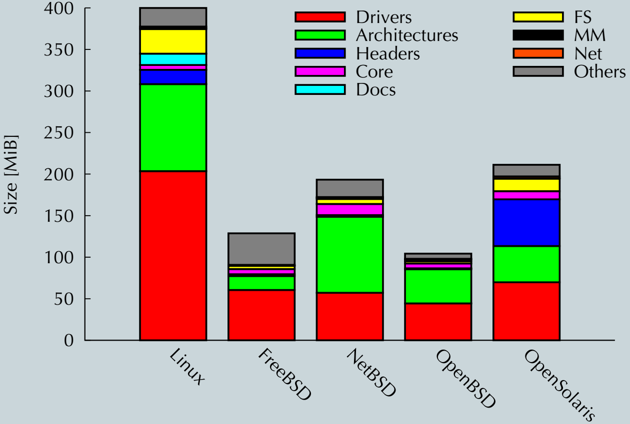
1. INTRODUCTION

Arguments regarding the efficacy of open
and development processes often employ
tributes [21], anecdotal evidence [17],
waving [13]. Although considerable
formed on open source artifacts and
9, 41, 3, 32], the direct comparis
ucts with corresponding propriet
an elusive goal. The recent ope
nel and the distribution of lar
source code to research inst
window of opportunity to
between the code of ope

Permission to make digital or hard copies of all or part of this wo
personal or classroom use is granted without fee provided that copies
not made or distributed for profit or commercial advantage and that cop
bear this notice and the full citation on the first page. To copy otherwise,
republish, to post on servers or to redistribute to lists, requires prior specific
permission and/or a fee.
ICSE'08, May 10–18, 2008, Leipzig, Germany.
Copyright 2008 ACM 978-1-60558-079-1/08/05 ...$5.00.

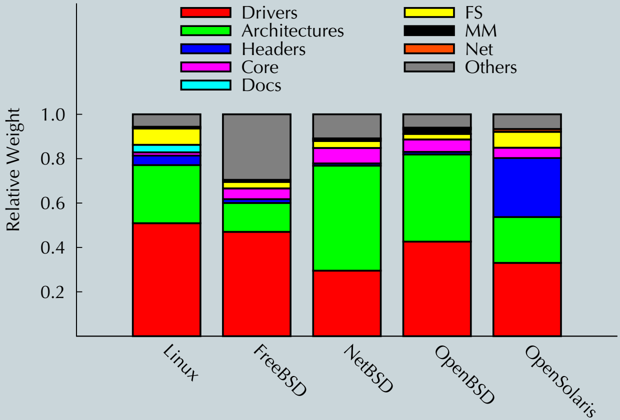
Code Shape

Code Shape: Composition



SIEMENS

Code Shape: Composition

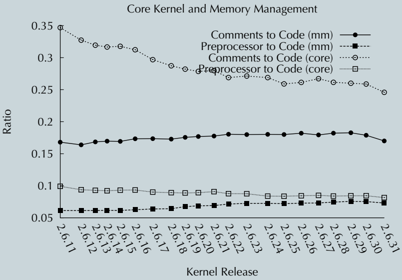


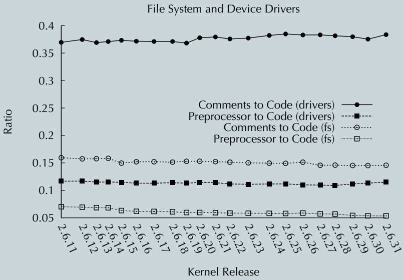
Code shape: Measurements

Measurements

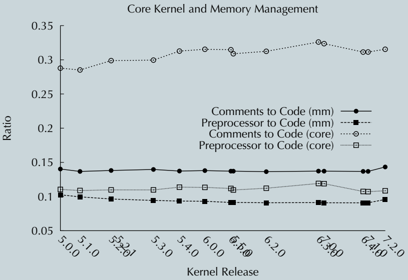
- Line counts for
  - Input
  - Blanks
  - Comments
  - Preprocessor Statements
  - Declarative/executable instructions
- Measured per
  - File
  - Function
  - Structure/Union
- Measured over $\approx$ 5 years of history

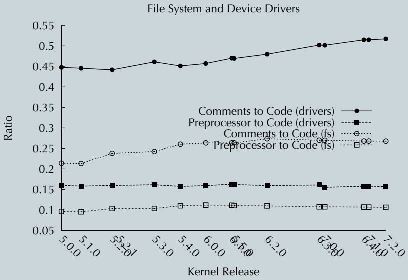
Code Shape: Linux





Code Shape: FreeBSD





Code Shape: Results

Pattern: Documentation

Memory Management $\lessapprox$ Filesystems $<$ Core Kernel $\ll$ Device Drivers

Pattern: Preprocessor Use

- Comparable between subsystems of kernels
- OpenBSD, NetBSD: More tendency towards preprocessor

Exception: OpenSolaris

- Very little preprocessor use
- Documentation: core $<$ fs $\lessapprox$ drivers $<$ mm

Code Shape: Results

Pattern: Documentation

Memory Management $\lessapprox$ Filesystems $<$ Core Kernel $\ll$ Device Drivers

Pattern: Preprocessor Use

- Comparable between subsystems of kernels
- OpenBSD, NetBSD: More tendency towards preprocessor

Exception: OpenSolaris

- Very little preprocessor use
- Documentation: core $<$ fs $\lessapprox$ drivers $<$ mm

Code Shape: Results

Pattern: Documentation

Memory Management $\lessapprox$ Filesystems $<$ Core Kernel $\ll$ Device Drivers

Pattern: Preprocessor Use

- Comparable between subsystems of kernels
- OpenBSD, NetBSD: More tendency towards preprocessor

Exception: OpenSolaris

- Very little preprocessor use
- Documentation: core $<$ fs $\lessapprox$ drivers $<$ mm

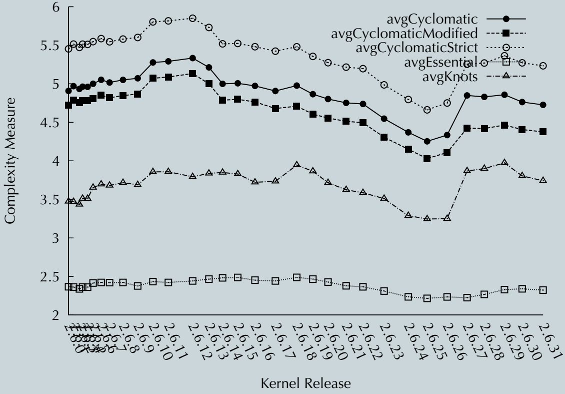
Complexity

Complexity Measures

Complexities

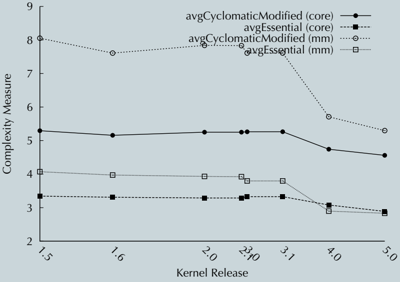
- Cyclomatic complexity
- Essential complexity
- Maximal nesting
- Maximal essential knots
- Henry-Kafura Information Flow

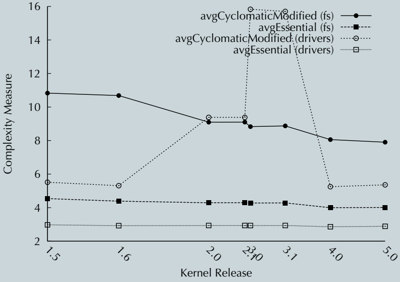
Complexity: Comparison of measures



- Similar picture for other systems
- *Long-term behaviour*, not absolute numbers important

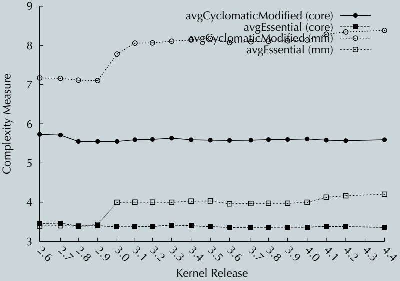
Complexity: NetBSD

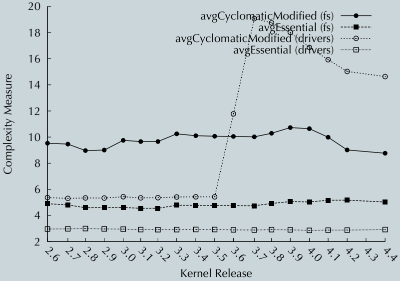




Discontinuities and Flips!

Complexity: OpenBSD





Discontinuities and Flips!

Complexity: Interpretation

Interpretation?

- Hard to attribute discontinuous jumps to specific causes
- Similar relations between subsystems
- Further averaging required

Complexity ordering

Let A, B denote subsystems. Define ordering (with $c \in]0.5, 1]$):

$$A \sqsubseteq B \Leftrightarrow \frac{1}{\#\mathcal{M} \cdot \#\mathcal{R}} \sum_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}} \Theta(m(B_r) - m(A_r)) \geq c$$

SIEMENS

Complexity: Interpretation

Interpretation?

- Hard to attribute discontinuous jumps to specific causes
- Similar relations between subsystems
- Further averaging required

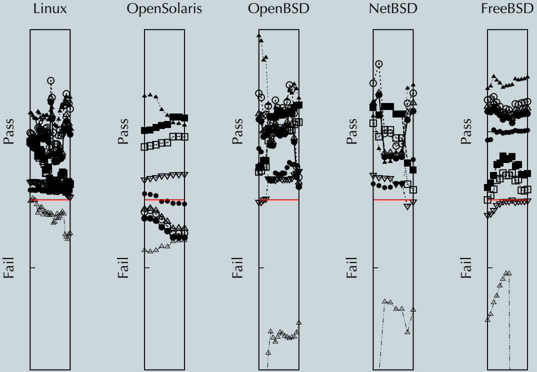
Complexity ordering

Let A, B denote subsystems. Define ordering (with $c \in]0.5, 1]$):

$$A \sqsubseteq B \Leftrightarrow \frac{1}{\#\mathcal{M} \cdot \#\mathcal{R}} \sum_{r \in \mathcal{R}} \sum_{m \in \mathcal{M}} \Theta(m(B_r) - m(A_r)) \geq c$$

SIEMENS

Complexity Ordering: Results



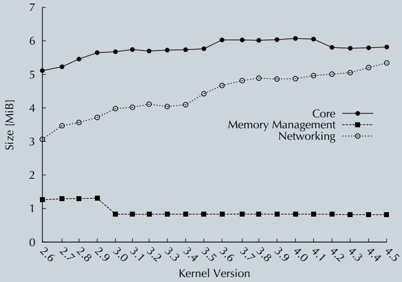
Hypothesis

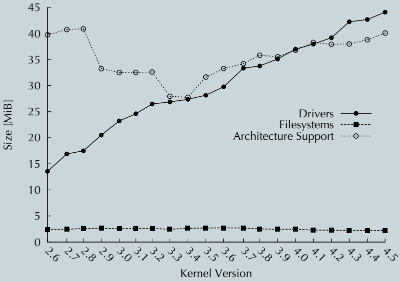
$$\text{Core} \sqsubseteq \text{MM} \wedge \text{MM} \sqsubseteq \text{FS}.$$

Size Matters

SIEMENS

Size Matters: OpenBSD



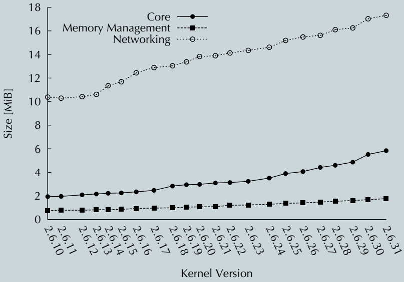


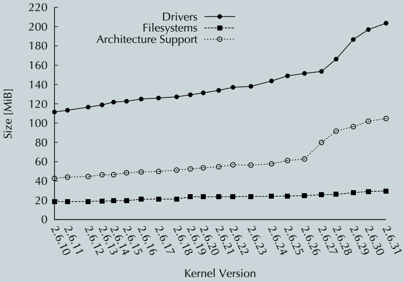
Results

- Discontinuities attributable to specific causes
- Per-subsystem consideration required
- Linear vs. superlinear

SIEMENS

Size Matters: Linux

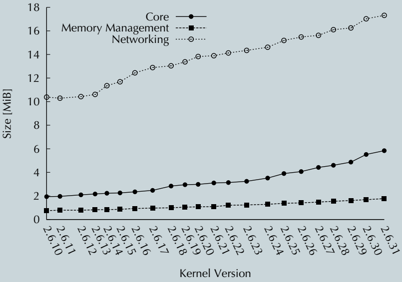


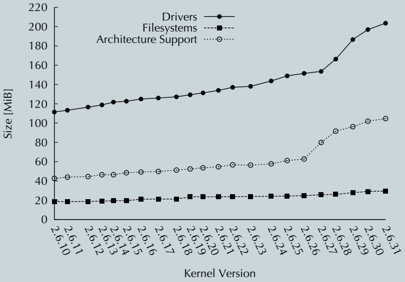


Results

- Discontinuities attributable to specific causes
- Per-subsystem consideration required
- Linear vs. superlinear

Size Matters





Results

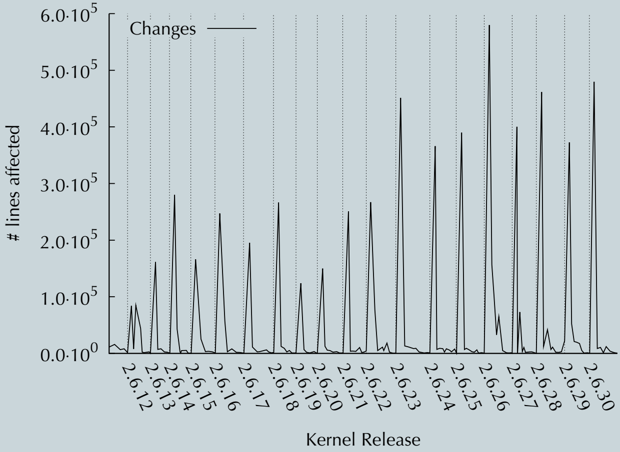
- Discontinuities attributable to specific causes
- Per-subsystem consideration required
- Linear vs. superlinear

Dynamics

Dynamics of development

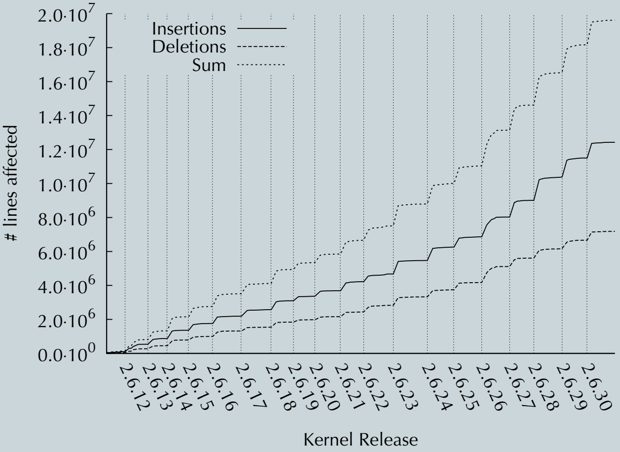
- Global analysis: Traditional method
- VCSs allow decomposition to patch level
- Good time resolution essential

Dynamics: Linux



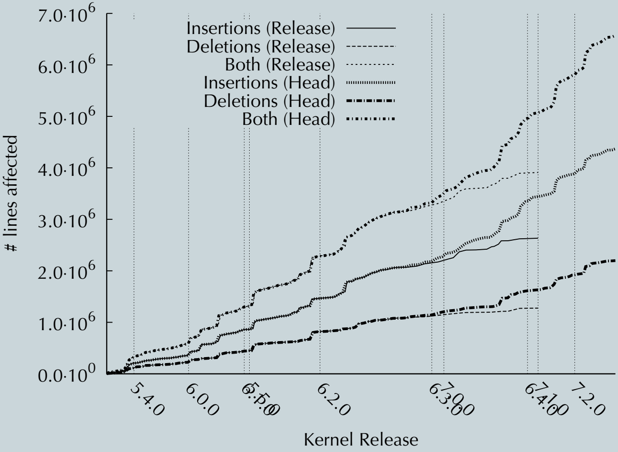
(Well known from lwn.net and the Linux foundation)

Dynamics: Linux

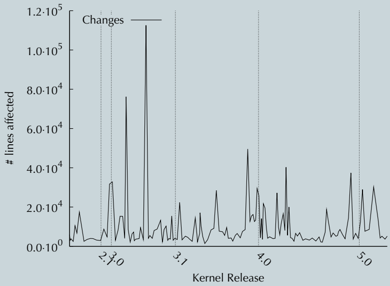


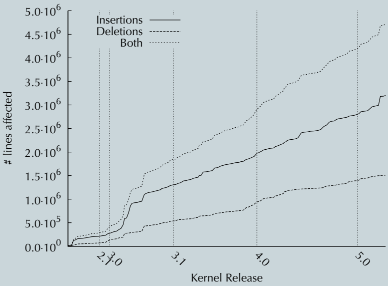
(Well known from lwn.net and the Linux foundation)

Dynamics: FreeBSD



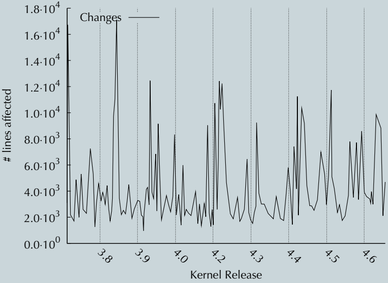
Dynamics: NetBSD

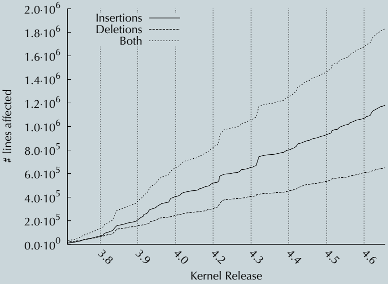




SIEMENS

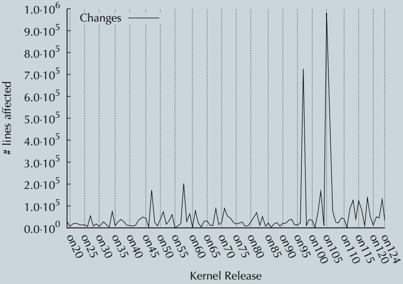
Dynamics: OpenBSD

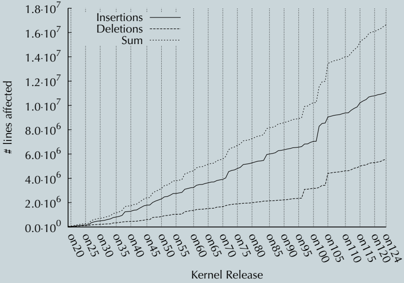




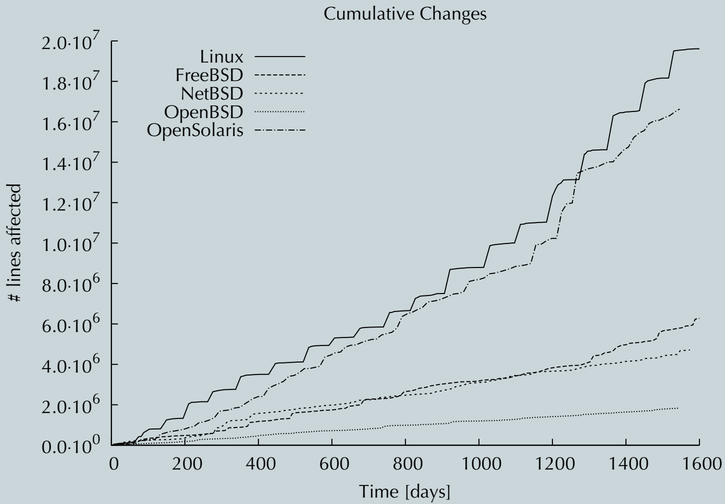
SIEMENS

Dynamics: OpenSolaris

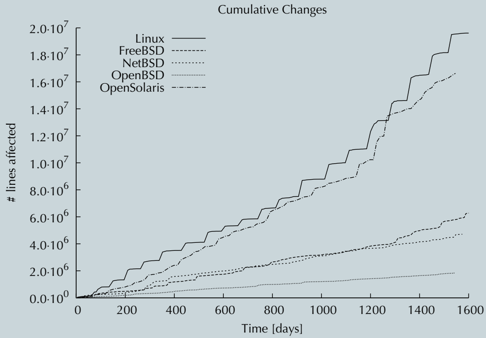




Dynamics: Comparison



Dynamics: Comparison



Conclusions

- VCS: Key influence on dynamics
- Linear vs. superlinear: Relevant?
- Maintainability criteria don't match reality

Outline

Summary

Single static snapshots ✗

Dynamics ✓
Metrics ✓ ✗

Version Control ✓
Maintainability Indices ✗

Quantitative similarities ✓

- Documentation, Preprocessor patterns
- Complexity relationships

Thank you for listening!