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

1 Introduction

2 Evaluated Systems

3 Feature Comparison

4 Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics

5 Summary
Outline

1  Introduction

2  Evaluated Systems

3  Feature Comparison

4  Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics

5  Summary
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?
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?
Outline

1. Introduction

2. Evaluated Systems

3. Feature Comparison

4. Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics

5. Summary
Evaluated Systems: Linux and OpenSolaris

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

- License: CDDL
- Version Control: Mercurial
The BSD “family”

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

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

- **OpenBSD**
  - License: BSD
  - Version Control: CVS
Outline

1. Introduction
2. Evaluated Systems
3. Feature Comparison
4. Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics
5. Summary
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
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
Outline

1. Introduction
2. Evaluated Systems
3. Feature Comparison
4. Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics
5. Summary
Quantitative measurements

Quantitative ≠ Useful

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

Question

Can measurements provide meaningful insights?
Quantitative measurements

Quantitative != Useful

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

Question

Can measurements provide meaningful insights?

ABSTRACT

The FreeBSD, eXBSD, Solaris, and Windows operating systems have kernels that provide comparable facilities. Interestingly, their code bases share almost no common parts, whereas 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.2.9 [Software Engineering]: Metrics—Process metrics

General Terms

Measurement

1. INTRODUCTION

Arguments regarding the efficacy of open source development processes often employ tributes [21], anecdotal evidence [17], or reasoning [15]. Although considerable research has been devoted to open source artifacts and their dependencies, a direct comparison of the direct consequences of the open source model is comparatively lacking. The recent salience and the distribution of the open source model in research and industry is a result of both an opportunity to research the consequences of the open source model between the code of open source projects and a lack of such research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ACM 08.05.08, 2008, Leipzig, Germany.
Copyright 2008 ACM 978-1-59593-978-7/08/0005...

W. Mauerer
Corporate Technology SE 2
Code Shape
Code Shape: Composition

The diagram shows the size distribution of different components across various operating systems. Each operating system is represented by a bar, and the components are color-coded as follows:

- Drivers: Red
- Architectures: Green
- Headers: Blue
- Core: Purple
- Docs: Cyan
- FS: Yellow
- MM: Black
- Net: Orange
- Others: Gray

The sizes are measured in MiB.
Code Shape: Composition

![Chart showing relative weight of different components in various operating systems.]

- Linux
- FreeBSD
- NetBSD
- OpenBSD
- OpenSolaris

Components:
- Drivers
- Architectures
- Headers
- Core
- Docs
- FS
- MM
- Net
- Others

Legend:
- Red: Drivers
- Green: Architectures
- Blue: Headers
- Pink: Core
- Cyan: Docs
- Yellow: FS
- Black: MM
- Orange: Net
- Gray: Others

Relative Weight:
- Linux: 0.2
- FreeBSD: 0.4
- NetBSD: 0.6
- OpenBSD: 0.8
- OpenSolaris: 1.0

Page 14 October 29, 2009 W. Mauerer Corporate Technology SE 2
Measurements

- Line counts for
  - Input
  - Blanks
  - Comments
  - Preprocessor Statements
  - Declarative/executable instructions

- Measured per
  - File
  - Function
  - Structure/Union

- Measured over ≈ 5 years of history
Code Shape: Linux

Core Kernel and Memory Management

- Comments to Code (mm)
- Preprocessor to Code (mm)
- Comments to Code (core)
- Preprocessor to Code (core)

File System and Device Drivers

- Comments to Code (drivers)
- Preprocessor to Code (drivers)
- Comments to Code (fs)
- Preprocessor to Code (fs)
Code Shape: FreeBSD
Code Shape: Results

Pattern: Documentation

Memory Management \( \preceq \) 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 \( \preceq \) drivers \( < \) mm
## Pattern: Documentation

Memory Management ≈ Filesystems < Core Kernel ≪ 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 ≈ drivers < mm
Code Shape: Results

Pattern: Documentation

Memory Management ≈ Filesystems < Core Kernel ≪ 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 ≈ drivers < mm
Complexity
# Complexity Measures

## Complexities

- Cyclomatic complexity
- Essential complexity
- Maximal nesting
- Maximal essential knots
- Henry-Kafura Information Flow
Complexity: Comparison of measures

- avgCyclomatic
- avgCyclomaticModified
- avgCyclomaticStrict
- avgEssential
- avgKnots

Similar picture for other systems

Long-term behaviour, not absolute numbers important
Complexity: NetBSD

Discontinuities and Flips!
Complexity: OpenBSD

Discontinuities and Flips!
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 $c \in ]0.5, 1]$):

$$A \sqsubseteq B \iff \frac{1}{\#M \cdot \#R} \sum_{r \in R} \sum_{m \in M} \Theta(m(B_r) - m(A_r)) \geq c$$
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 $c \in ]0.5, 1]$):

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

Hypothesis

\[ \text{Core} \sqsubseteq \text{MM} \land \text{MM} \sqsubseteq \text{FS}. \]
Size Matters
Size Matters: OpenBSD

Results

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

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

![Graph showing the dynamics of FreeBSD over different kernel releases.](image)
Dynamics: NetBSD

Kernel Release

Changes

# lines affected

Insertions

Deletions

Both

# lines affected
Dynamics: OpenBSD

![Graph showing changes, insertions, deletions, and both over kernel releases.]

- **Changes**
- **Insertions**
- **Deletions**
- **Both**

Kernel Release:

- $0.0 \cdot 10^0$
- $2.0 \cdot 10^5$
- $4.0 \cdot 10^5$
- $6.0 \cdot 10^5$
- $8.0 \cdot 10^5$
- $1.0 \cdot 10^6$
- $1.2 \cdot 10^6$
- $1.4 \cdot 10^6$
- $1.6 \cdot 10^6$
- $1.8 \cdot 10^6$
- $2.0 \cdot 10^6$

# lines affected

- $3.8$
- $3.9$
- $4.0$
- $4.2$
- $4.3$
- $4.4$
- $4.5$
- $4.6$
Dynamics: OpenSolaris

# lines affected
Kernel Release
Changes

Insertions
Deletions
Sum

0.0⋅10^0
2.0⋅10^6
4.0⋅10^6
6.0⋅10^6
8.0⋅10^6
1.0⋅10^7
1.2⋅10^7
1.4⋅10^7
1.6⋅10^7
1.8⋅10^7

0.0⋅10^0
2.0⋅10^6
4.0⋅10^6
6.0⋅10^6
8.0⋅10^6
1.0⋅10^7
1.2⋅10^7
1.4⋅10^7
1.6⋅10^7
1.8⋅10^7
Dynamics: Comparison

Cumulative Changes

- Linux
- FreeBSD
- NetBSD
- OpenBSD
- OpenSolaris

# lines affected vs Time [days]

- Linux
- FreeBSD
- NetBSD
- OpenBSD
- OpenSolaris
Conclusions

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

1. Introduction

2. Evaluated Systems

3. Feature Comparison

4. Quantitative Measurements
   - Code Shape
   - Complexity
   - Size Matters
   - Dynamics

5. Summary
<table>
<thead>
<tr>
<th>Single static snapshots</th>
<th>Dynamics</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>✓</td>
<td>✓ X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Version Control</th>
<th>Maintainability Indices</th>
<th>Quantitative similarities</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓ X</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Documentation, Preprocessor patterns
- Complexity relationships
Thank you for listening!