

# 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?



## 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?~~

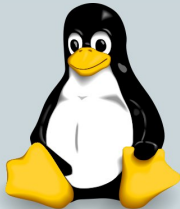


# Outline

- 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



- 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

- 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?

### A Tale of Four Kernels

Dionidis Spinellis  
Department of Management Science and Technology  
Athens University of Economics and Business  
Patission 76, GR-104 24 Athens, Greece  
dds@aubg.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 the 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 widely 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—Product metrics

#### General Terms

Measurement

#### 1. INTRODUCTION

Arguments regarding the efficacy of open source development processes often employ quantitative evidence [2], anecdotal evidence [37], or writing [2]. Although considerable research has focused on open source artifacts as a whole [9, 41, 3, 12], the direct comparison of artifacts with corresponding proprietary counterparts is an elusive goal. The recent open source movement and the distribution of open source code to research institutions has opened a window of opportunity to compare the code of open

source code with that of large industrial-scale open source code. This paper presents a significant analysis between four large open source operating systems. An additional open source code system, written in C, is also included to provide a baseline for development

2

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 made or distributed for profit or commercial advantage and that the fee code for copying and distribution appears on the first page. This permission is granted without fee for non-profit organizations and individuals. To request other permissions, contact the author.

© 2006 ACM 978-1-60558-079-1/06/0010...\$5.00  
Copyright 2006 ACM 978-1-60558-079-1/06/0010...\$5.00

# 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

Dionidis Spinellis  
Department of Management Science and Technology  
Athens University of Economics and Business  
Patission 76, GR-104 24 Athens, Greece  
dds@aubg.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 the 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 widely 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—Product metrics

#### General Terms

Measurement

#### 1. INTRODUCTION

Arguments regarding the efficacy of open source development processes often employ quantitative evidence [2], anecdotal evidence [37], or writing [2]. Although considerable research has focused on open source artifacts as a whole [9, 41, 3, 12], the direct comparison of artifacts with corresponding proprietary counterparts is an elusive goal. The recent open source movement and the distribution of free and open source code to research institutions has provided a window of opportunity to compare the quality between the code of open source and proprietary software.

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 made or distributed for profit or commercial advantage and that they bear this notice and the full citation on the first page. Copyrights are retained by the author(s). To request more information, contact the publisher at the address below.

10818-1, May 2006, 40 pages.  
Copyright 2006 ACM 978-1-60558-079-1/06/05...\$5.00.

Here I report on code quality metrics for large industrial-scale open source systems: OpenSolaris, and the Windows operating system. I analyze the contribution of the source code to the overall size of the system between four large open source systems. An additional goal is to compare the code quality of the source code systems with the code quality of the proprietary systems. I also compare the code quality of the source code systems with the code quality of the proprietary systems.

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

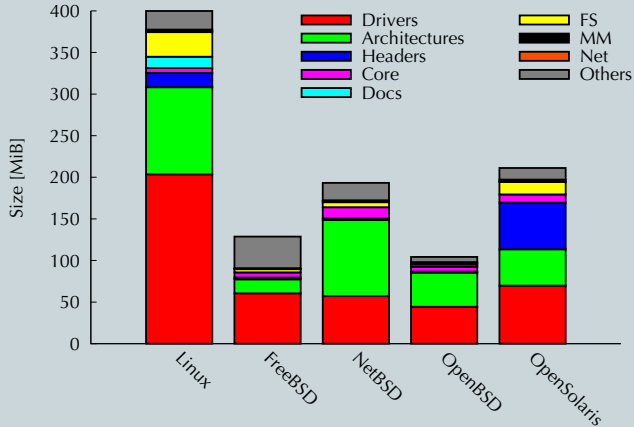
266

267

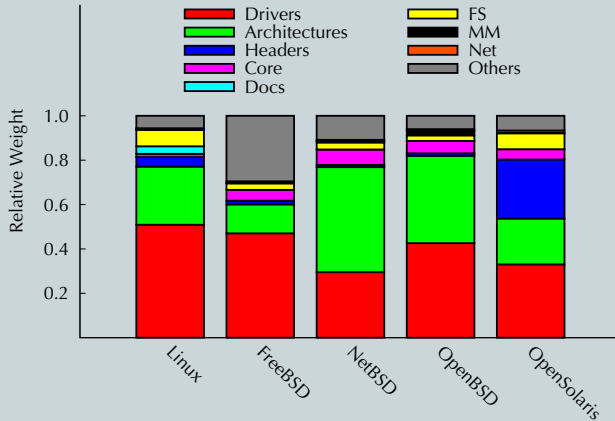
268

# Code Shape

# Code Shape: Composition



# Code Shape: Composition



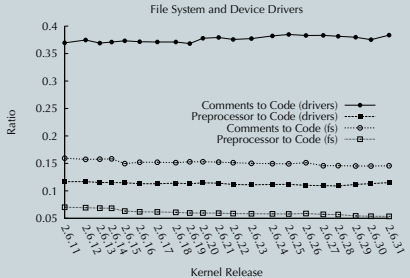
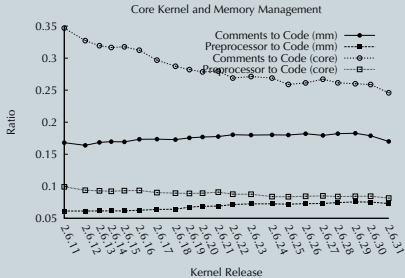


## Code shape: Measurements

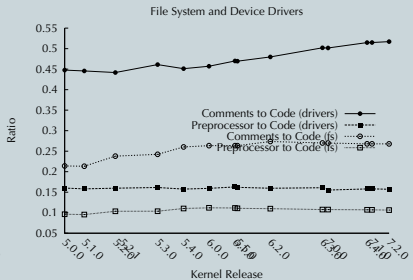
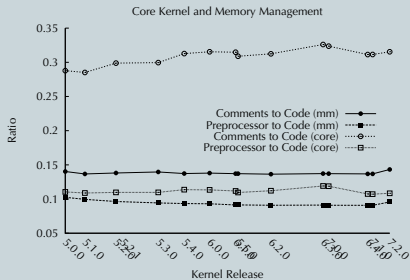
### Measurements

- 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  $\approx$  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  $\approx$  drivers  $<$  mm

## Code Shape: Results

### Pattern: Documentation

Memory Management  $\approx$  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  $\approx$  drivers  $<$  mm

## Code Shape: Results

### Pattern: Documentation

Memory Management  $\approx$  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  $\approx$  drivers  $<$  mm

# Complexity

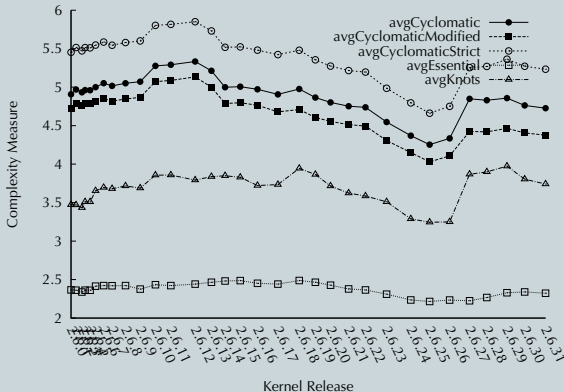
# Complexity Measures

## Complexities

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

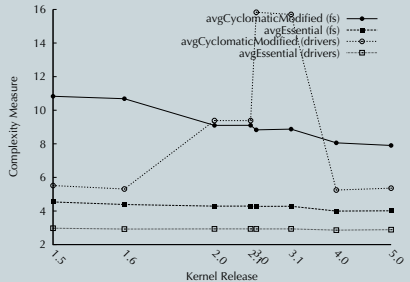
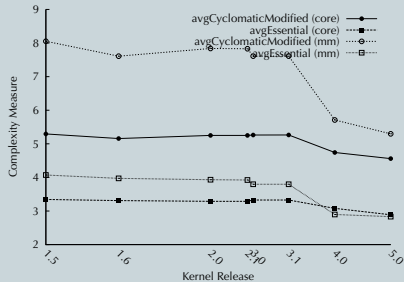


# Complexity: Comparison of measures



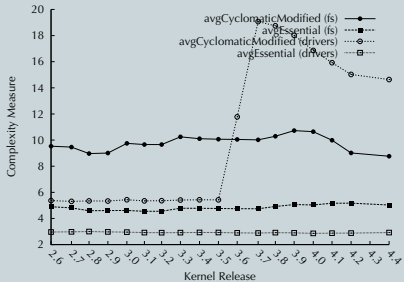
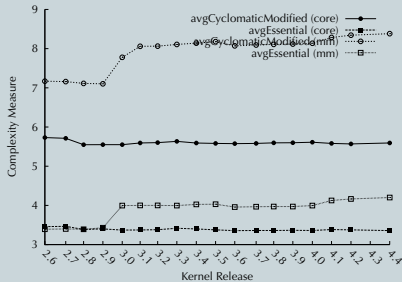
- Similar picture for other systems
- *Long-term behaviour, not absolute numbers important*

# 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  $c \in ]0.5, 1]$ ):

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

## Complexity: Interpretation

### Interpretation?

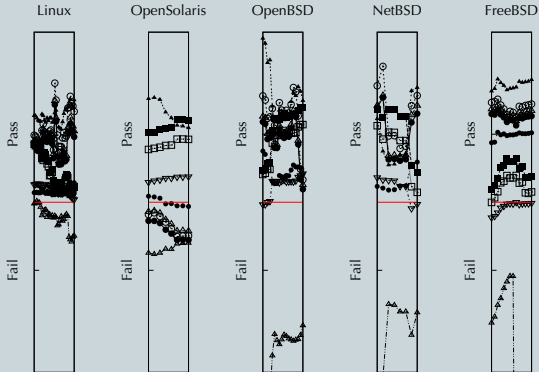
- Hard to attribute discontinuous jumps to specific causes
- Similar relations between subsystems
- Further averaging required

### 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

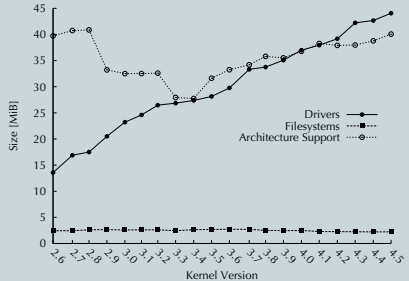
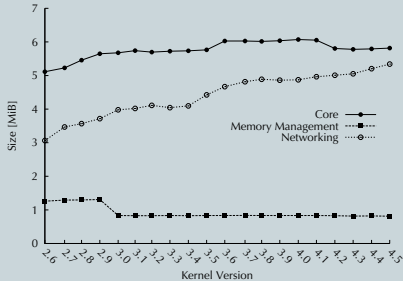


Hypothesis

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

# Size Matters

# Size Matters: OpenBSD



## Results

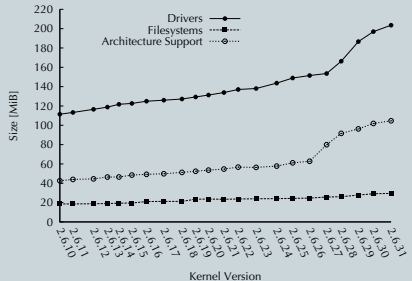
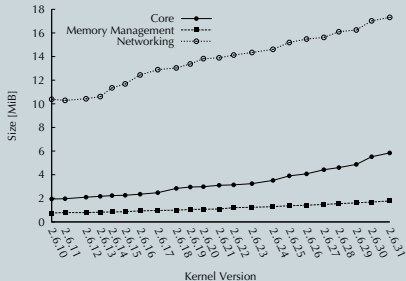
Discontinuities attributable to specific causes

Per-subsystem consideration required

Linear vs. superlinear



# Size Matters: Linux



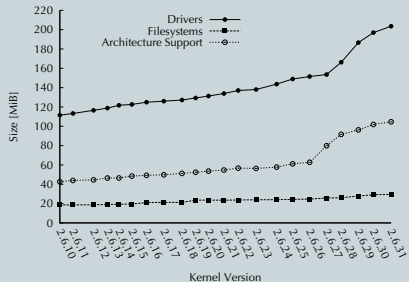
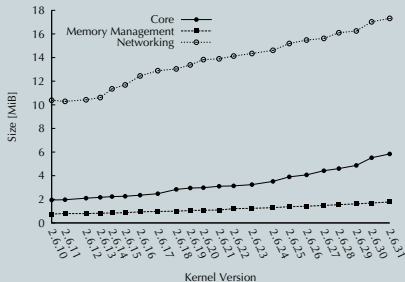
## Results

Discontinuities attributable to specific causes

Per-subsystem consideration required

Linear vs. superlinear

# Size Matters



## Results

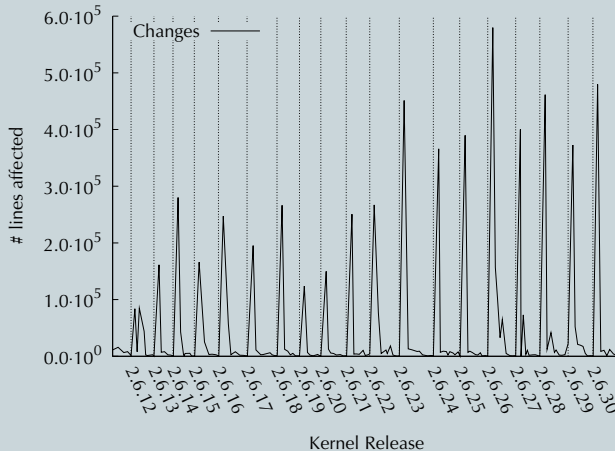
- Discontinuities attributable to specific causes
- Per-subsystem consideration required
- Linear vs. superlinear

# Dynamics

## Dynamics of development

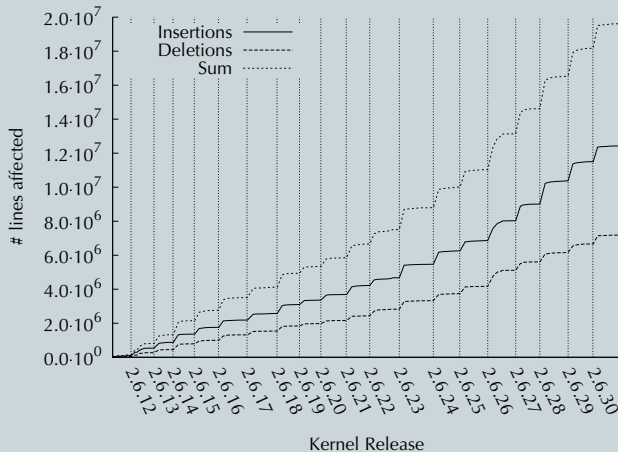
- Global analysis: Traditional method
- VCSs allow decomposition to patch level
- Good time resolution essential

# Dynamics: Linux



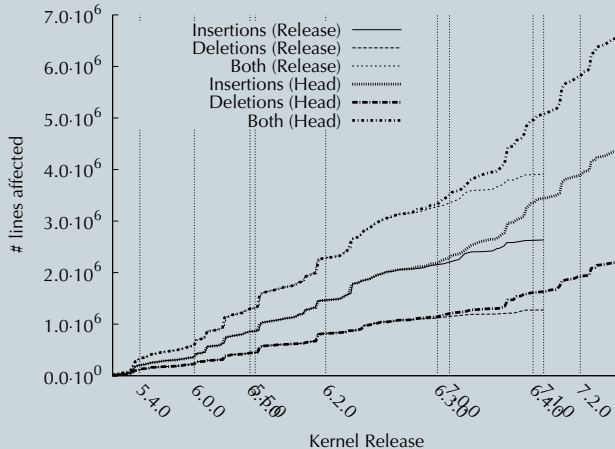
(Well known from lwn.net and the Linux foundation)

# Dynamics: Linux

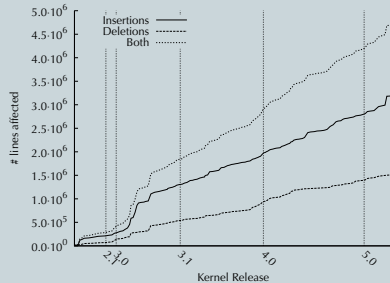
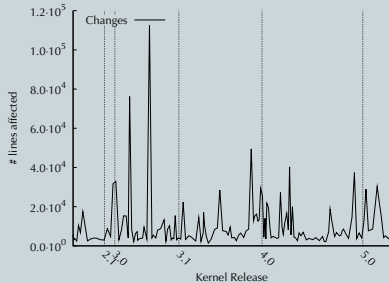


(Well known from lwn.net and the Linux foundation)

# Dynamics: FreeBSD

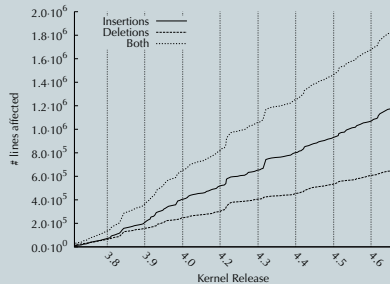
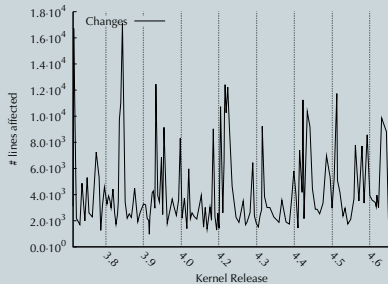


# Dynamics: NetBSD

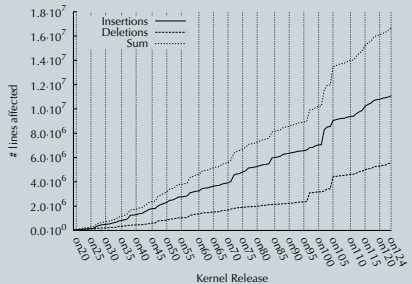
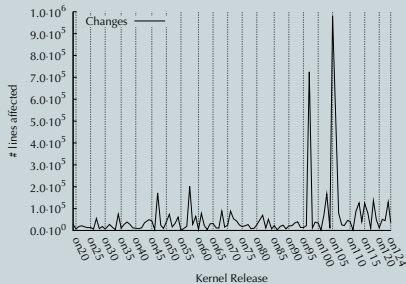




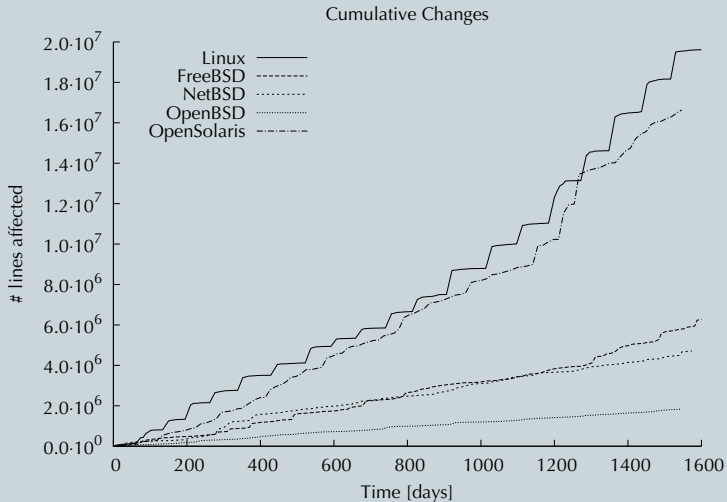
# Dynamics: OpenBSD



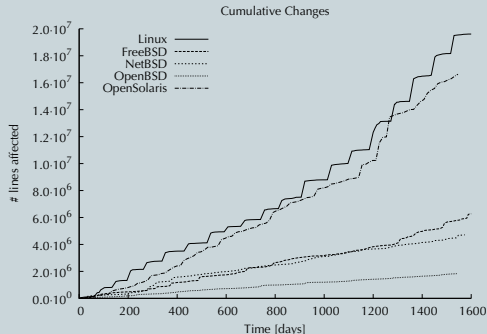
# Dynamics: OpenSolaris



# Dynamics: Comparison



## Dynamics: Comparison



## 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**


## Summary

Single static snapshots 

Dynamics 

Metrics  

Version Control   
Maintainability Indices 

Quantitative similarities 

- Documentation, Preprocessor patterns
- Complexity relationships

# Thank you for listening!