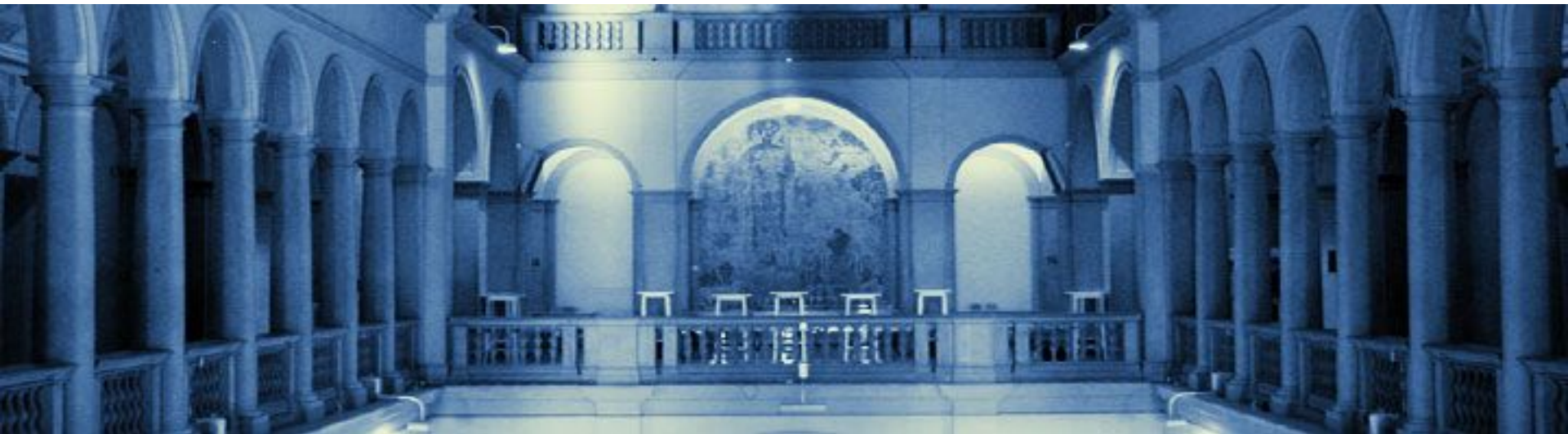


Code Reuse, Motivation, Koordination, Kollaboration: Was Entwicklerteams von Open Source Communities lernen können

Matthias Stürmer, ETH Zürich, mstuermer@ethz.ch, www.stuermer.ch
Internet Briefing Entwickler Konferenz 2008, 28. Mai 2008, Zürich



Überblick

- Warum von Open Source Communities lernen?
- Datenquellen: >30 Experten-Interviews
- Lernen von Open Source Communities
 1. Code Reuse
 2. Motivation
- Schlussfolgerungen

Matthias Stürmer

- BWL und Informatik, Universität Bern
- Doktorand ETH Zürich
- nice – TYPO3 Websites und Web Applikations-Entwicklung
- /ch/open
 - OpenExpo
 - informatica08-Initiative Hackontest
 - Open Source Software an Schulen
 - Ubuntu Swiss Remix

Warum von Open Source Communities lernen?

- Grosser Einfluss von OSS auf proprietäre Software
- Evolutionäre, bottom-up Prozesse
- Dezentral & effizient komplexe Software erstellen
- Hochqualifizierte Entwickler motivieren

→ Aber auch OSS kann von herkömmlicher Software-Entwicklung lernen!

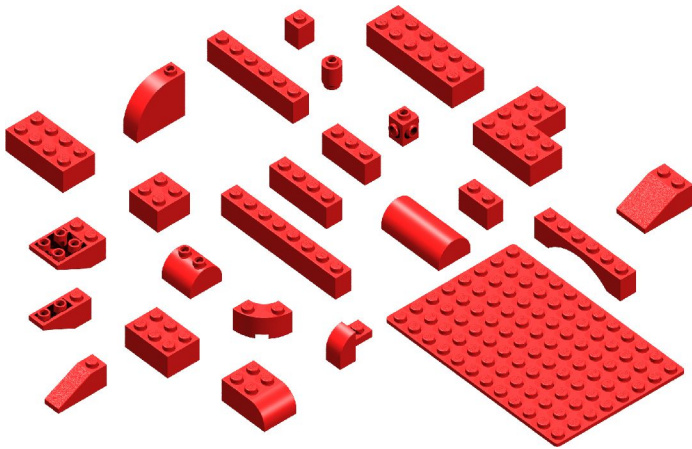
Datenquellen: >30 Experten-Interviews

- 10 Interviews mit **Komponenten**-Entwicklern
 - libpng, zlib, libxml, expat etc.
 - Working Paper „Learnings from Software Reuse in OSS“
- 23 Interviews mit Nokia Managern, Entwicklern, Nokia-Partnerfirmen und **Maemo**-Community
 - Working Paper zu Nutzen und Kosten von OSS
- Interviews mit **OpenMoko**-Schlüsselpersonen
 - Sean Moss-Pultz und Michael Lauer



Lernen von Open Source Communities

Lernfeld 1: Code Reuse



- Effizienzmerkmal
- Modularität ist Voraussetzung für Code Reuse
- OSS ist modularer als proprietäre Software (MacCormack et al., 2006)
- Hohe Wiederverwendung in OSS mittels Libraries (Haefliger et al., 2008)

Modulares vs. monolithisches Design

■ Vorteile von Modularität

- Aufgabentrennung; Verbindung über Schnittstellen
- Dezentrale Entwicklung möglich
- Bessere Skalierbarkeit/Erweiterbarkeit
- Bessere Wartbarkeit

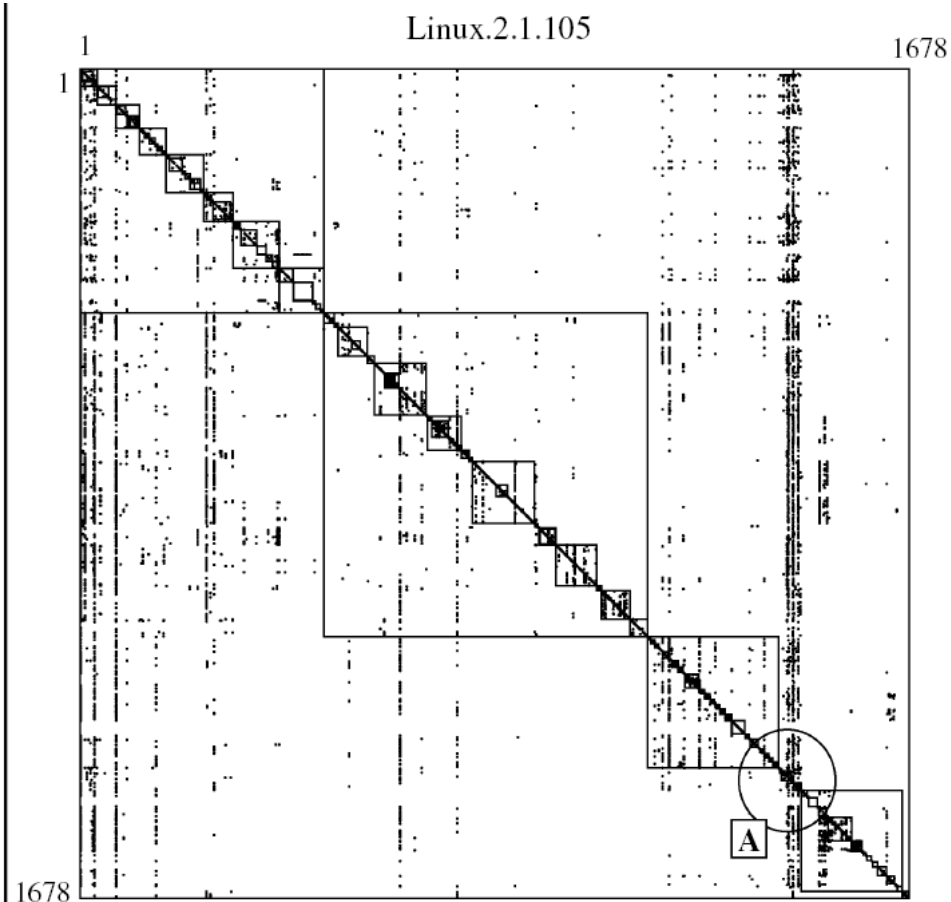
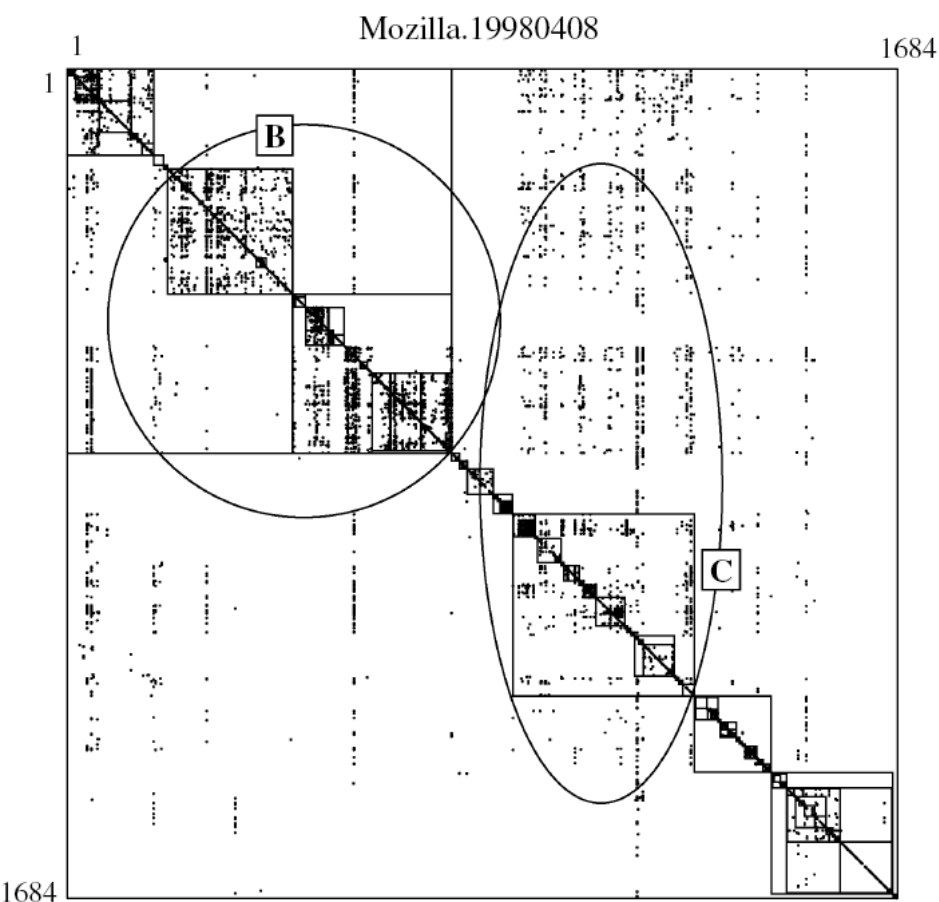
■ Umsetzungsebenen

1. Kern mit Plug-Ins, Extensions, Add-ons...
2. Basisbibliotheken und End-User Applikationen

Quelle: MacCormack et al., 2006

Mozilla vs. Linux

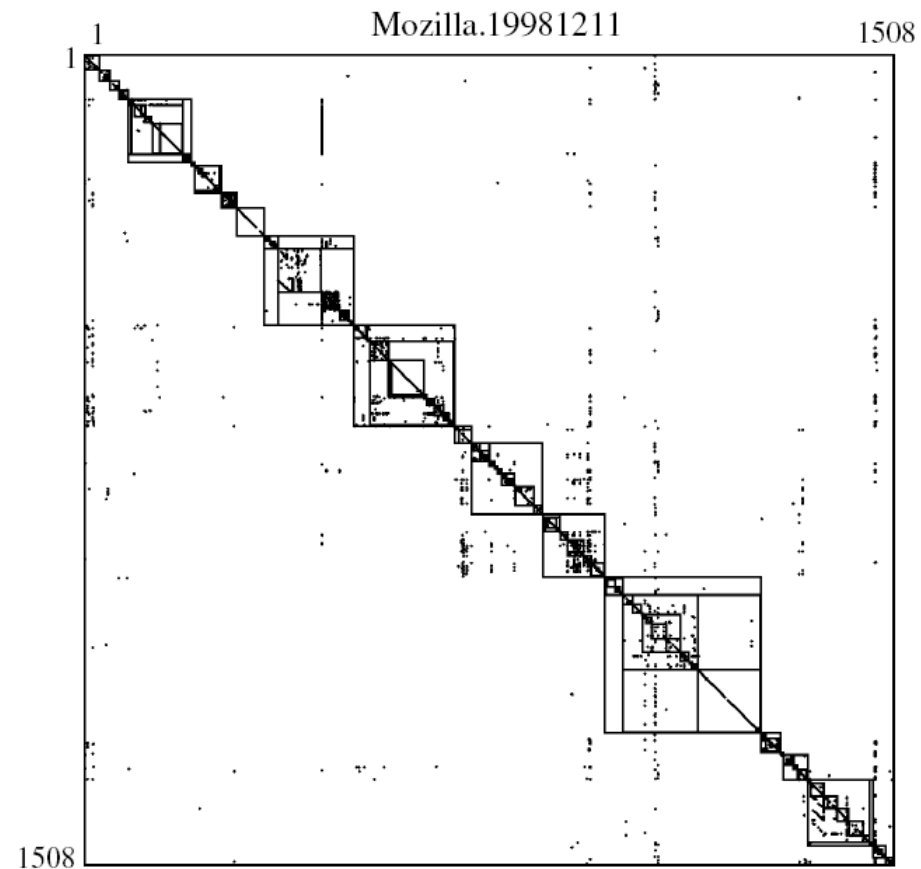
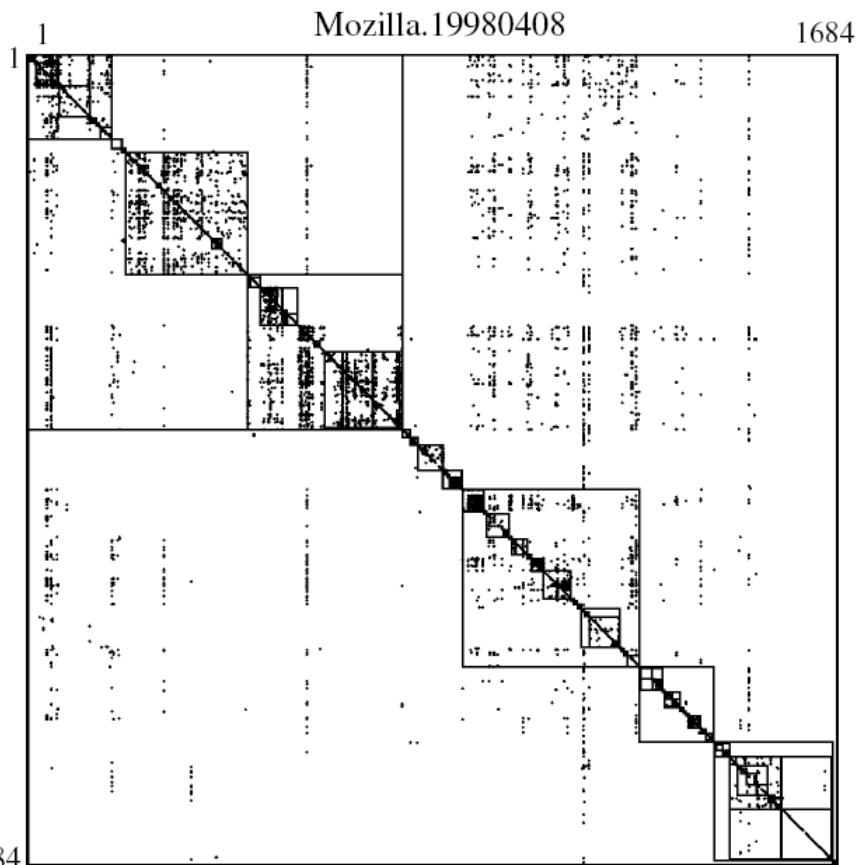
DSM: Design Structure Matrix



Quelle: MacCormack et al., 2006

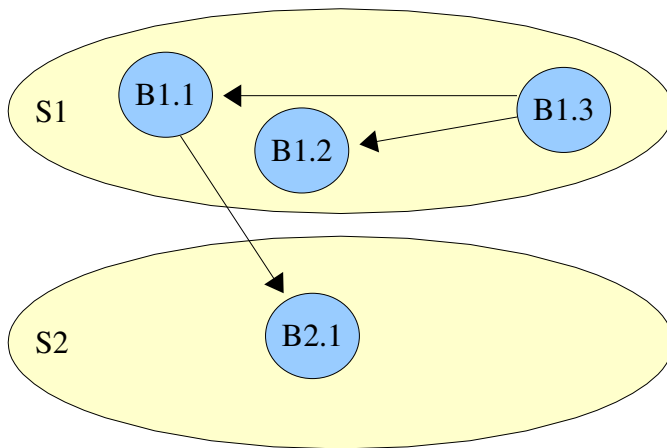
Mozilla vorher vs. Nachher

DSM: Design Structure Matrix



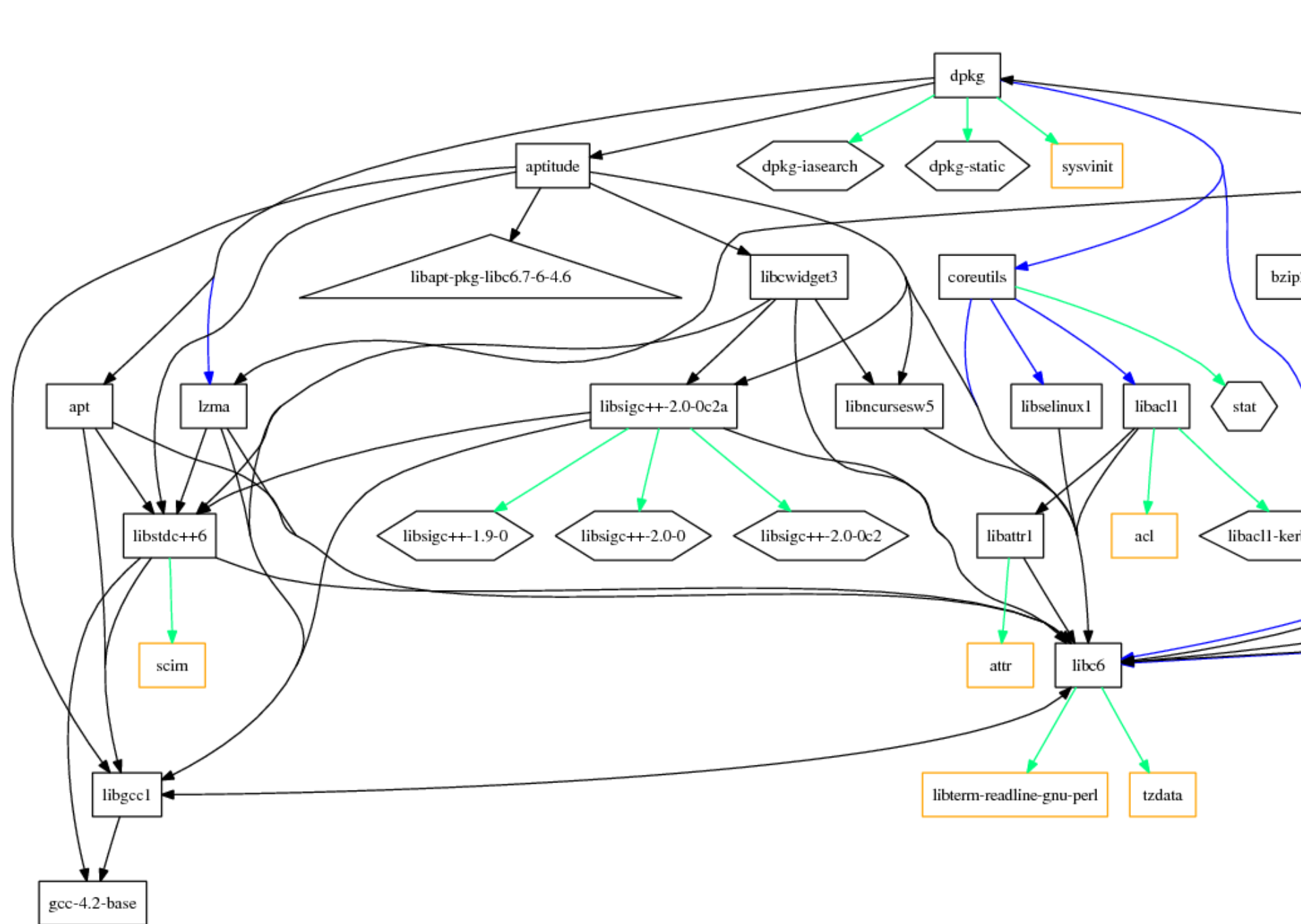


Modularität und Code Reuse in Debian

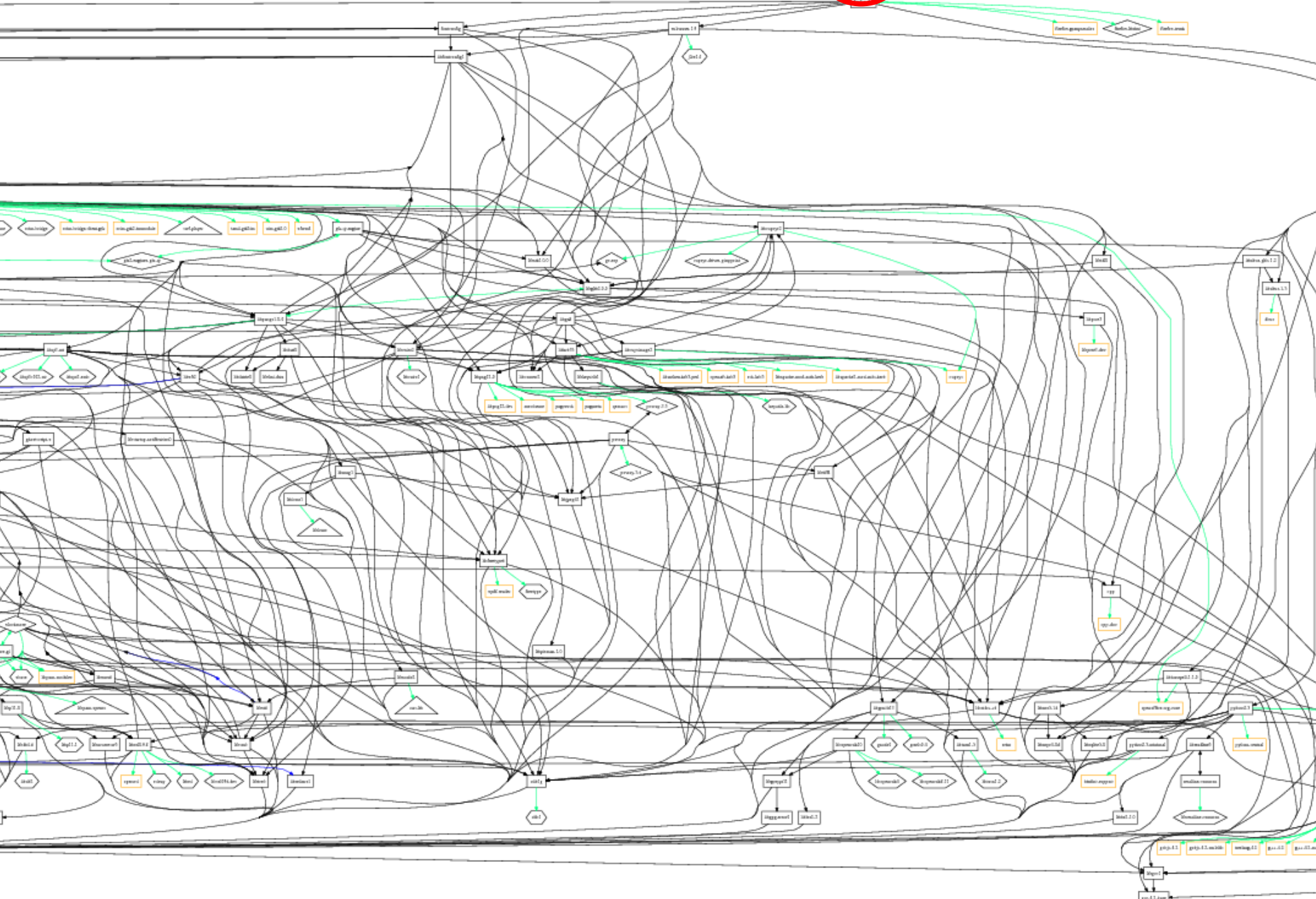


- Abhängigkeiten bei binären Paketen
- Binäre Pakete kompiliert aus Source Code Paketen
- Hohe Wiederverwendung (Ø 36x)

Quelle: Spaeth et al., 2008a



Mozilla Firefox



Wie erreicht man mehr Code Reuse?

1. Software Architektur noch **modularer**

- Voraussetzung für Code Reuse, in Unix „Do one thing, do it well.“

2. **Kultur** von Code Reuse fördern

- NIH Syndrom verhindern, „Reuse or make decision“, Ownership

3. Quellcode eigener Bibliotheken [intern] **freigeben**

- Beiträge anderer, höhere Qualitätsansprüche, stärkere Diffusion

4. Selber **Beitragen** bei anderen Bibliotheken

- „Stilles“ Wissen erhalten und Software-Wartung outsourcen
- Reputations-Denken und Mitwirkung an OSS fördern (upstream)

Lernfeld 2: Motivation



- Intrinsische vs. extrinsische Motivation
- Quellcode öffnen
- Interner Wettbewerb
- Community Building

Motivationen von Open Source Entwicklern

- **Intrinsische Motivation** (=aus dem Arbeitsinhalt)
 - Basierend auf Spass an der Arbeit
 - Basierend auf Verpflichtungsgefühl
 - Förderbar durch Selbstbestimmung und Wertschätzung
- **Extrinsische Motivation** (=externer Anreiz)
 - Nicht-monetär: Ansehen, Karrierechancen, Ferien etc.
 - Monetär: Anstellungsvertrag, Bonus, Aufträge etc.
 - Anreize sind Präferenzen-abhängig

Offenheit motiviert zu höherer Code-Qualität

- „At work, the programming I do is not as good as I do for something like Expat. At work the requirements is that it needs to get out there, it needs to get working fast. It just needs to work, it doesn't matter how good it is, doesn't matter how elegant, you know. [...] **For Open Source I do much more thinking, I have much more desire to do it properly.**“

expat-Bibliothek Entwickler

Gesunder Wettbewerb

- **In Open Source Communities**
 - KDE vs. GNOME, Joomla! vs. TYPO3, XML Libraries...
 - Z.T. Zusammenarbeit auf Komponenten-Ebene
- **Wettbewerbsebenen**
 - Extern: Konkurrenz-Firmen
 - Intern: Entwickler-Teams
- **Coopetition**
 - cooperation (Kooperation)
 - competition (Wettbewerb)

Community Building

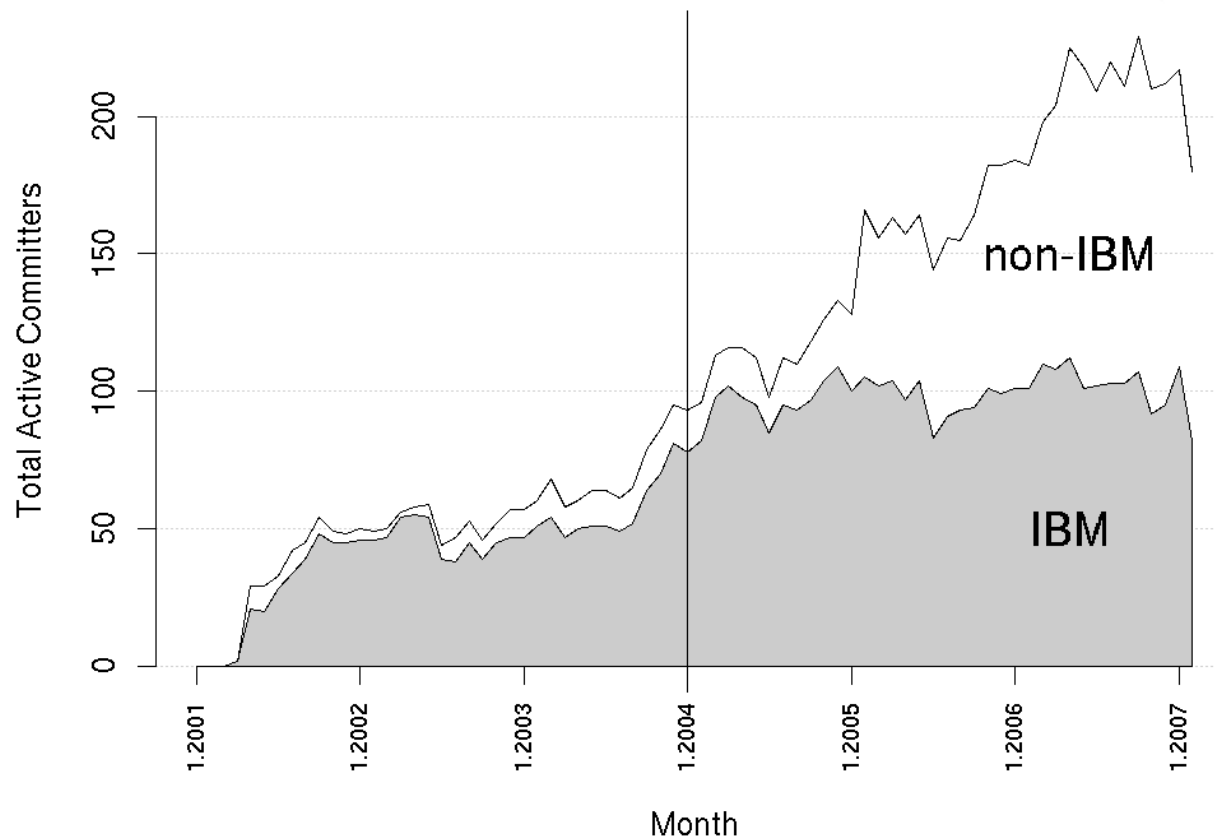
■ User Innovation

- Software-Anwender tragen zur Weiterentwicklung bei
- Testen, Fehlerbeschreibungen, Dokumentation etc.
„Many eyes make bugs shallow“ (Raymond, 1999)

■ Open Innovation

- Externe Entwickler motivieren mitzuwirken
- Selber Software freigeben

Beispiel 1: Eclipse



Quelle: Spaeth et al., 2008b

Lines of Code in Eclipse von 2001 bis 2007

IBM vs. nicht-IBM Entwickler

	LOCs before Foundation (4.01–12.03: 33 months)		LOCs after Foundation (1.04-2.07: 37 months)		Total LOCs (4.01-2.07: 70 months)		
IBM	8,410,972	73.94%	31,346,054	60.42%	39,757,026	62.85%	313 committers
Non-IBM	2,042,365	17.95%	19,507,145	37.60%	21,549,510	34.07%	252 committers
Unidentified	922,488	8.11%	1,027,085	1.98%	1,949,573	3.08%	40 committers
Total	11,375,825	100.00%	51,880,284	100.00%	63,256,109	100.00%	605 committers

COCOMO: 21.5 Millionen LOC externe Beiträge

~ 214,000 Mann-Monate ~ 1.7 billion USD

Beispiel 2: Nokia Internet Tablets



- Lanciert 2005
- Kein GSM
- Eigenes Debian
- Modelle bis heute:
 - N770
 - N800
 - N810

Community Building by Nokia

- Verkauf von **1000 vergünstigten Tablets** für OSS Entwickler
- **maemo.org** für Tutorials, Roadmap, API Doc, Wiki, Blog Planet...
- 595 registrierte **Maemo-Projekte** auf garage.maemo.org [28.5.08]
- **Mailing Lists** (Juni 2005 - Dezember 2006) und IRC Chat
 - Developer: 6795 Mails von 832 Email-Adressen (79 Nokia)
 - User: 2534 Mails von 511 Email-Adressen (33 von Nokia)
- Bugzilla für **Bug Reporting**: 3142 Bugs [28.5.08]
- Maemo **Software Development Kit**
- **Sardine**: Development (unstable) Version des Betriebssystem



Schlussfolgerungen

Fazit Code Reuse

1. Software Architektur noch **modularer**
2. **Kultur** von Code Reuse fördern
3. Quellcode eigener Bibliotheken [intern] **freigeben**
4. Selber **Beitragen** bei anderen Bibliotheken

Fazit Motivation

1. Ganzheitliche Motivationsförderung

- Autonomie bei Technologieentscheidungen
- Konstruktives Feedback, eigene Ziele setzen
- Nicht-monetäre Belohnungen wie Konferenzbesuche

2. Quellcode intern bzw. extern zeigen

3. Wettbewerb zwischen Entwicklerteams fördern

4. Software-Anwender in Entwicklung einbeziehen

Potential von Open Source Entwickler

- OSS Communities gewöhnt an dezentrale, kollaborative Arbeit und offene Kommunikation
- Somit: Erfahrene OSS Entwickler anstellen
 - **Hard Skills**
gute Programmierer, gute Software-Reuser
 - **Soft Skills**
Zusammenarbeit, Selbstverantwortung, Motivation

Referenzen

Haefliger, S.; von Krogh, G. & Spaeth, S. (2008) "Code Reuse in Open Source Software" Management Science, 54, 180-193

MacCormack, A.; Rusnak, J. & Baldwin, C. Y. (2006) "Exploring The Structure Of Complex Software Designs: An Empirical Study Of Open Source And Proprietary Code" Management Science, 52, 1015-1030

Michlmayr, M. & Senyard (2006) „A Statistical Analysis of Defects in Debian and Strategies for Improving Quality in Free Software Projects“ in The Economics of Open Source Software Development, A. Bitzer, J. & Schröder, P. J. H. (ed.) Elsevier B.V., 131-148

Raymond, E. S. (1999) "The Cathedral & the Bazaar" 1st Edition. O'Reilly, Sebastopol, CA,

Sebastian Spaeth, Georg von Krogh, Matthias Stuermer, Stefan Haefliger (2008a) "A Lightweight Model of Component Reuse: A Study of Software Packages in Debian GNU/Linux" working paper

Sebastian Spaeth, Matthias Stuermer, Georg von Krogh (2008b) „Enabling Knowledge Creation through Outsiders: Towards a Push Model of Open Innovation“ working paper

Stuermer, M. (2005). 'Open Source Community Building', master thesis, University of Bern, <http://opensource.mit.edu/papers/sturmer.pdf>

Diskussion

Fragen, Bemerkungen, Anregungen?

Matthias Stürmer
mstuermer@ethz.ch
www.stuermer.ch