Open Source Development in a Nutshell

Andreas Bauer

baueran@in.tum.de

“Perlen der Weisheit”

Der Linux Siegeszug

Was macht den augenscheinlichen Erfolg dieses Betriebssystems aus?

- Technische Überlegenheit?
- Kostenfaktor?
- Lösung bisher ungelöster Probleme?
- Befriedigung von Benutzerwünschen? (Treiber, Bedienbarkeit, usw.)
- …

Open-Source Charakter

Basiert auf den 4 Freiheiten:

- Freiheit (1) das Programm für jeden Zweck einzusetzen
- Freiheit (2) den Code zu lesen und an die eigenen Bedürfnisse anzupassen
- Freiheit (3) den Code an andere weiterzugeben
- Freiheit (4) den Code zu verbessern und andere daran teilhaben zu lassen ($\rightarrow$ 3)

Achtung: es gibt auch “Freibier-Code”.

Entwicklungsmodell

- Über das Internet verteilt
- Weltweit tausende Entwickler (noch mehr Anwender)
- Dynamischer, “schneller” und öffentlicher Prozess
- *Project leader: skills over authority*
- Resultiert in robusten Anwendungen

$\Rightarrow$ Führt zwangsläufig zu verzerrten Vorstellungen darüber, wie OSS eigentlich funktioniert.

Vorurteile

- Chaos-Theorie
- Unendliche Ressourcen an Freizeitprogrammierern
- Mindere Qualität
- Kommerziell irrelevant
- Kein Support
- Keine Standards
- “Es geht immer nur um MicroSoft”
- …

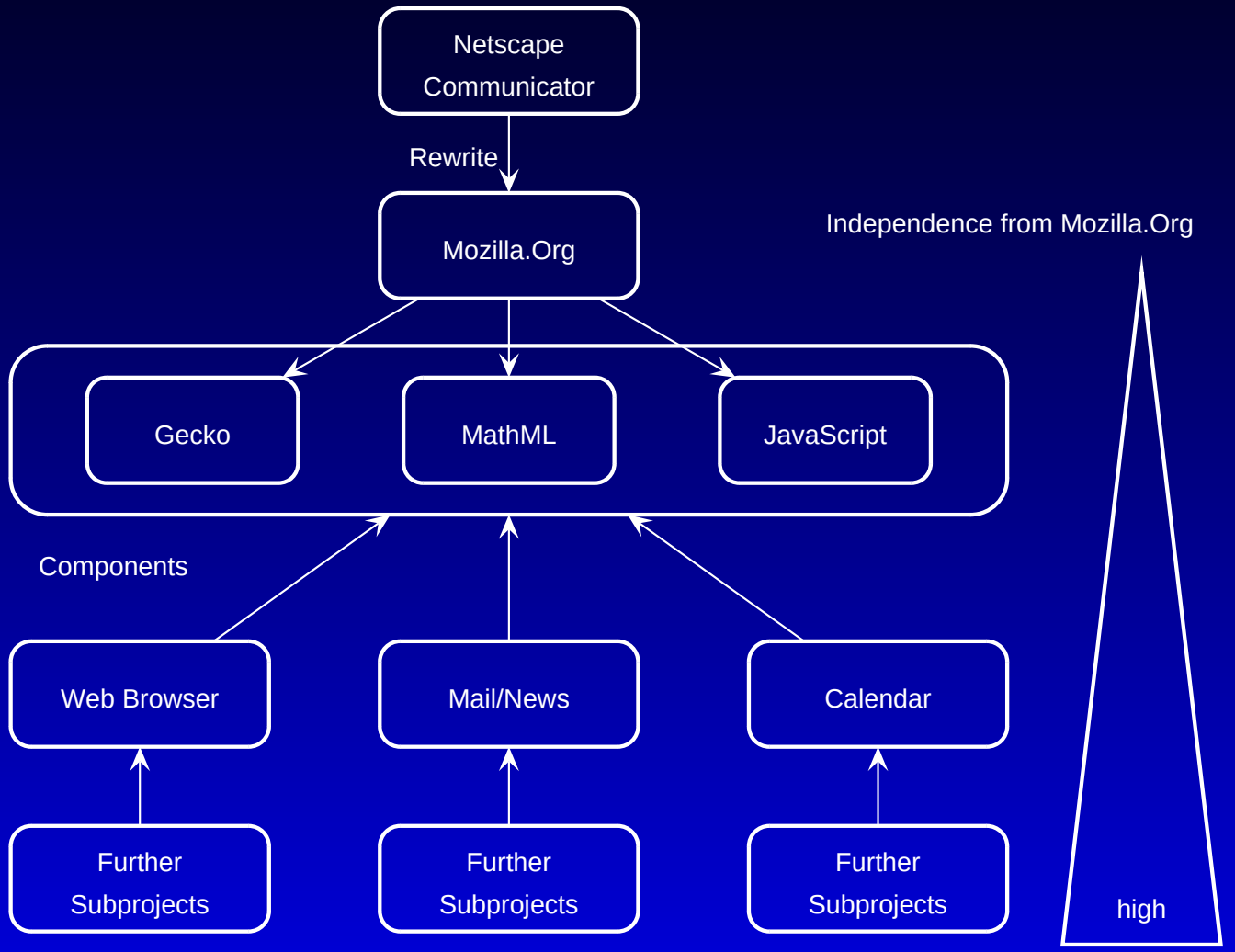
Wie funktioniert OSS?

- Anfangs mit fast keinem admin. Overhead:
  - Ein Prototyp wird erstellt
  - Eine Website wird eingerichtet die Entwickler ziehen soll (nicht Anwender!)
  - Das Produkt V1.0 wird schließlich erstellt
- Später, Kommunikation mittels Mailing-Listen und öffentl. Code Reviews
- Bei großen Projekten: kleine, fokussierte “Core-Teams”, mehr Politik
- Projektleiter normalerweise der beste, nicht der lauteste!

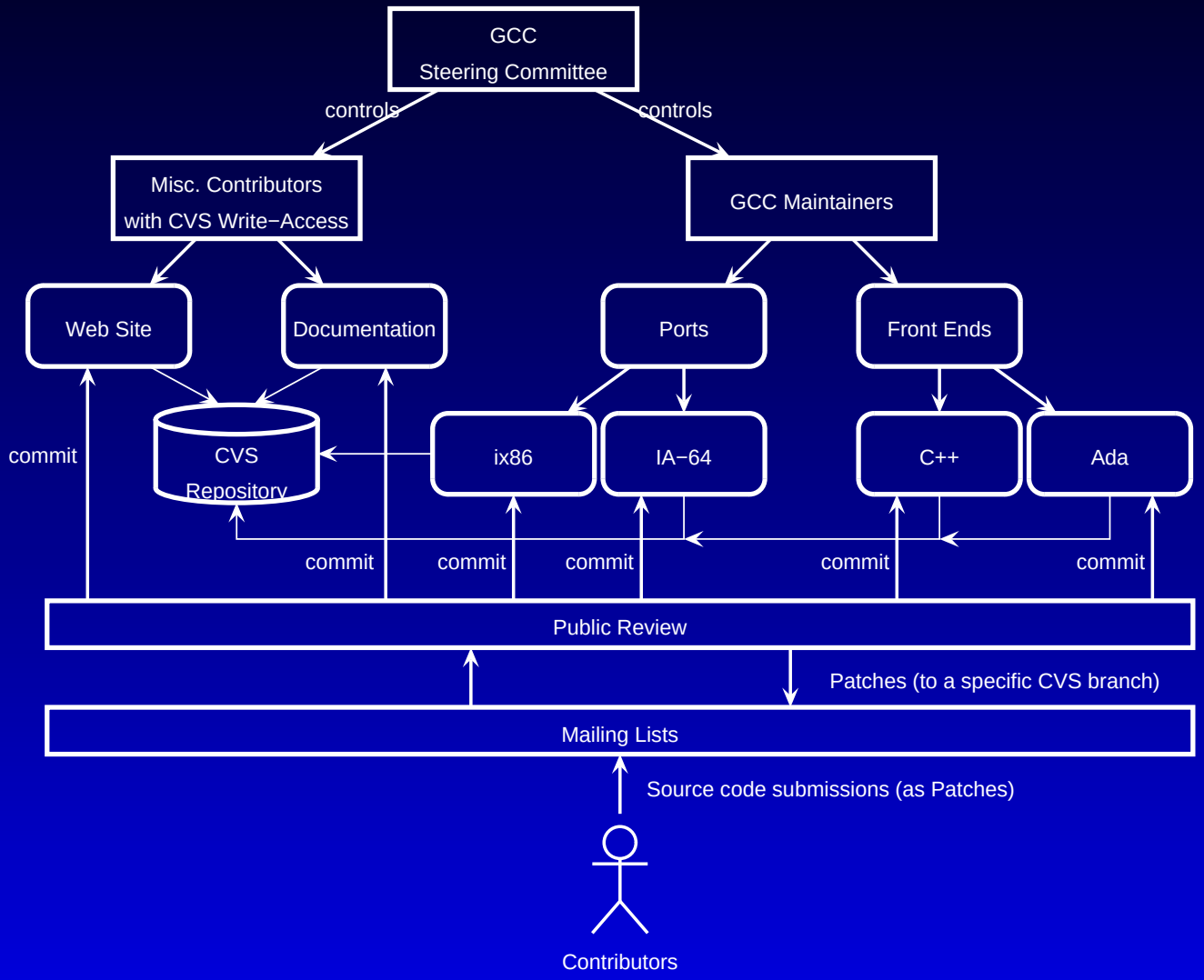
Wie funktioniert's? (#2)

- Planung:
  - Aus der Not eine Tugend machen: programmiert wird was notwendig ist
  - **Tool-Unterstützung:** die Tools entwickeln sich parallel mit den Projektanforderungen weiter
- **Evolution:**
  - guter Code überlebt, schlechter wird neugeschrieben (konventionell ein Disaster!)
  - Projektaufteilung in handhabbare Einheiten (Mozilla, GCC, etc.)
  - Organisation und Architektur sind eng verkoppelt

Mozilla


Netscape Communicator
Rewrite
Mozilla.Org
Gecko
MathML
JavaScript
Components
Web Browser
Mail/News
Calendar
Further Subprojects
Further Subprojects
Further Subprojects
Independence from Mozilla.Org
high

GCC


GCC
Steering Committee
controls
controls
Misc. Contributors
with CVS Write−Access
GCC Maintainers
Web Site
Documentation
Ports
Front Ends
commit
CVS
Repository
ix86
IA−64
C++
Ada
commit
commit
commit
commit
commit
Public Review
Patches (to a specific CVS branch)
Mailing Lists
Source code submissions (as Patches)
Contributors

Und wie nicht?

Konventionelles Vorgehen:

- Formale Spezifikation (diese soll endgültig sein)
- Design-Spezifikation
- Design der Schnittstellen
- Evtl. Marketing
- Programmierung in kleinen Teams
- Aufteilung oft nach geographischer Lage, anstatt von Kompetenzen
- Hierarchische Struktur
- Kaum direkte Kommunikation...
- ...aber viel verlorene Zeit in z.B. Meetings

OSS Nachteile

Um OSS-Entwicklung positiv (z.B. durch Tools) zu beeinflussen, müssen die Defizite klar sein:

- Code wird nicht nur ersetzt, sondern oft auch dupliziert
- Schnittstellen ändern sich oft
- OSS ist meistens nicht dokumentiert
- Einstieg nicht immer einfach

Fragen?