„The Cathedral & the Bazaar“

von Eric S. Raymond

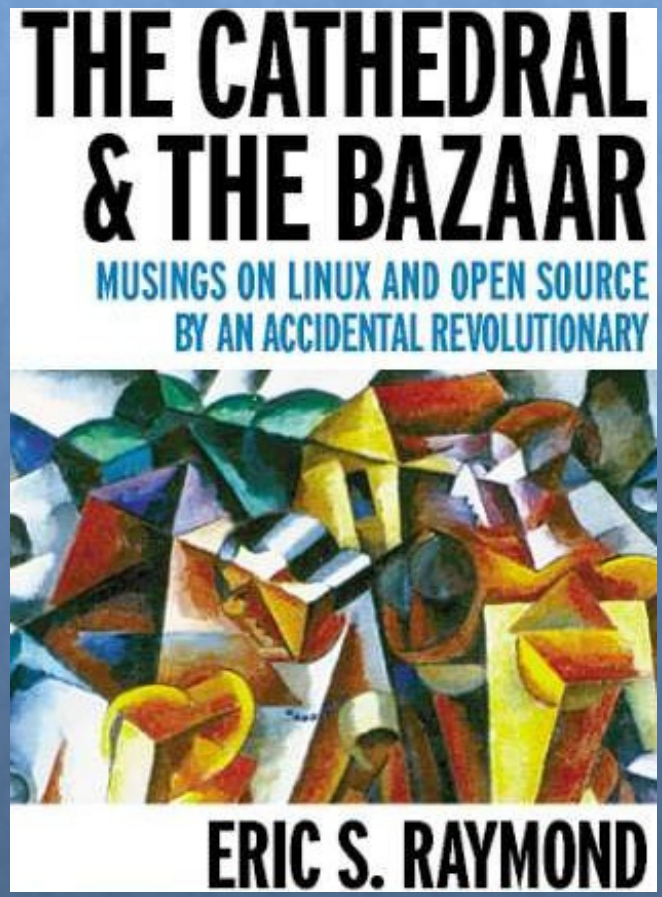
Inhalt

1. Eric S. Raymond
2. Das Kathedralen- und Basar-Modell
   - 2.1. Das Kathedralen-Modell
   - 2.2. Das Basar-Modell
3. Ausgangssituation
4. Programmieren nach dem Basar-Modell
5. Fazit

ERIC S. RAYMOND

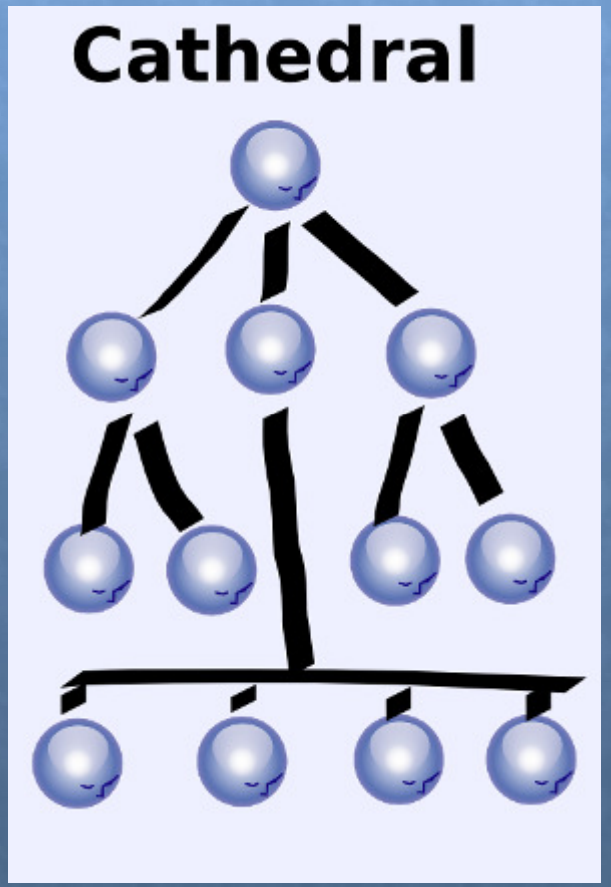
- US-amerikanischer Autor und Programmierer in der Hacker- und Open-Source-Szene
- führt ca. 30 Open Source-Projekte, verfasst Bücher und ist Weblogger
- Mitbegründer der Open Source Initiative (OSI)
- Seit Ende der 1990er wurde Raymond zu einer der bekanntesten und umstrittensten Figuren der Open Source-Bewegung.
- The Cathedral & The Bazaar:

  2 Programmierungsmodelle werden vorgestellt



Das Kathedralen-Modell

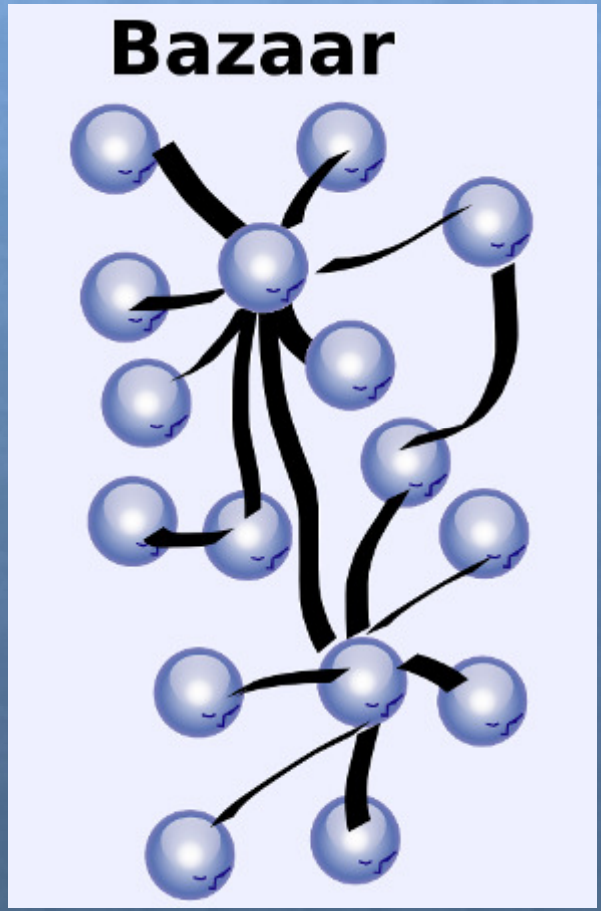
- Der Quellcode eines Programms wird nicht veröffentlicht
- In den Entwicklungszeiträumen wird der Quellcode nur von einer Entwicklergruppe programmiert
- Das Programm wird wie eine Kathedrale gebaut: Ein Chefarchitekt überwacht eine hierarchisch organisierte Gruppe von eingeweihten Spezialisten
- Keine Beta-Version
- Zentralistisch
- A priori



Das Basar-Modell

- der Quellcode ist in jedem Stadium über das Internet einsehbar
- Marktstände = User
- User finden/suchen nach Problemen und verbessern diese

→ Erfolg durch vermeintlich chaotische Selbstorganisation



Linux als ein Beispiel für das Basar-Modell:

„Die Linux-Gemeinde scheint ein großer, wild durcheinander plappernder Basar von verschiedenen Zielsetzungen und Ansätzen zu sein, der nur durch eine Reihe von Wundern ein kohärentes und stabiles System hervorbringen kann.“

AUSGANGSSITUATION

- Raymond arbeitete lange Zeit mit dem Kathedralen-Modell

  → Ein neues Projekt sollte mit dem Basar-Modell realisiert werden

  (praktische Überprüfung der These, dass das Basar-Modell funktioniert)

- es ging um die direkte Übermittlung von e-Mails zwischen

  verschiedenen Servern

- Mithilfe von diesem Projekt entwickelte Raymonds 19 Richtlinien, wie

  eine gute Open-Source-Software programmiert werden sollte

Programmieren nach dem Basar-Modell

1. **Jede gute Software beginnt mit den persönlichen Sehnsüchten eines Entwicklers.**

- *Sprichwort „Not macht erfinderisch“*

2. Gute Programmierer wissen, welchen Code sie schreiben sollen. Großartige Programmierer wissen, welchen Code sie umschreiben (und recyclen) können.

- Gute Programmierer sind faul
- Die Wiederverwertung von bereits vorhandenen Codes („reusing of codes and ideas“)
- Linux arbeitet nur mit offenen Quellen, die jedem zugänglich sind

5. Wenn du das Interesse an einem Programm verlierst, ist es deine Pflicht, dieses einem kompetenten Nachfolger zu übergeben.

- Sobald man merkt, dass man das Interesse am Programmieren verliert, sollte man es an jemanden weitergeben, den es interessiert

6. Die Anwender als Mit-Entwickler zu sehen ist der Weg zu schnellen Verbesserungen und Fehlerbehebungen, der die geringsten Umstände macht.

- Viele User ermöglichen es schnell Fehler zu finden und zu korrigieren

7. Veröffentliche früh. Veröffentliche häufig. Und höre auf die Benutzer.

- *Zentrale Idee des Basar-Modells*

8. Mit einer hinreichend großen Gruppe von Betatestern und Mit-Entwicklern wird fast jedes Problem schnell erkannt und die Lösung von jemandem gefunden.

- Für das Basar-Modell ist es von großer Wichtigkeit viele User zu haben (User = Mitentwickler)

11. Das zweitbeste nach eigenen guten Ideen ist das Erkennen von guten Ideen von Benutzern. Manchmal ist letzteres sogar das bessere.

- Die meisten Mit-Entwickler sind Hacker, die sich mit Open-Source-Programmen auskennen
- Leitmotiv für das Basar-Modell

12. Meist entstehen die brillanten Lösungen aus der Erkenntnis, dass das Problem falsch verstanden wurde.

- Soll nicht zögern, überholte Features aufzugeben, wenn die Effektivität davon unbeeinflusst bleibt.

13. „Perfektion (im Design) ist nicht erreicht, wenn man nichts mehr hinzufügen kann, sondern wenn nichts mehr entfernt werden kann.“

- Benutzerfreundlichkeit der Programme / Codes

19. Mit genügend guter Kommunikation, wie über das Internet, und Führung ohne Zwang sind viele Köpfe immer besser als einer.

- Der Leiter eines Basar-Projekts muss mit Leuten umgehen und kommunizieren können
- Die vorderste Front der Open Source-Software von Leuten geschaffen werden, deren individuelle Weitsicht und Brillanz dann durch die effektive Konstruktion von Gemeinden von Freiwilligen mit ähnlichen Interessen verstärkt wird

MANAGEMENT

- Klassisches Management ist bei Open-Source eher „unpassend“
- → Open Source-Entwickler arbeiten freiwillig. Sie sind nach Interesse und Fähigkeit selbsternannt, um an einem Projekt mitzuwirken
- Das stärkste Argument für Open Source ist die dezentralisierte, unentwegte Kritik und Verfeinerung durch Gleichgesinnte die allen konventionellen Methoden der Qualitätssicherung weit überlegen sind.

FAZIT

- Open-Source-Software, die mithilfe des Basar-Modell s programmiert wird, ist, obwohl sie vermeintlich chaotisch aussieht, eine sehr gute Alternative zum Kathedralen-Modell

Voraussetzungen:

1. Großer Userstamm
2. Effizientes Arbeiten
3. Schnelles Release
4. Für alle öffentlich zugänglich

Diskussion

Welche Vorteile hat das Kathedralen-Modell?

Für welche Arten von Software könnte das Kathedralen-Modell sinnvoller sein?

Wie steht es um die Sicherheit bei Open-Source-Software?