14. Oktober 2014

Open-Source-Softwareentwicklung

Wie schreibe ich ein FLOSS Programm?

UNI BASEL

Inhalt

- **„The Art of Unix Programming“, die 17 Regeln**
- **„The Cathedral and the Bazaar“, die 19 Richtlinien**
- **Truck Numbers**

Die 17 Regeln aus „The Art of Unix Programming“

#1 Modularität

- Schreibe einfache Bestandteile, die durch saubere Schnittstellen verbunden werden.
- Versuche die globale Komplexität niedrig zu halten.

Die 17 Regeln aus „The Art of Unix Programming“

#2 Klarheit

- Klarheit ist besser als Gerissenheit.
- Erste Priorität beim Programmieren sollte sein dem Leser des Codes zu erklären was der Code überhaupt macht.
- Vermeide es für kleine Performanceverbesserungen die lesbarkeit des Codes zu opfern.

Die 17 Regeln aus „The Art of Unix Programming“

#3 Aufbau

- Entwirf Programme so, dass sie mit anderen Programmen verknüpft werden können.
- Dafür sollte das Programm unabhängig sein.

Die 17 Regeln aus „The Art of Unix Programming“

#4 Trennung

- Trenne den Grundgedanken von der Umsetzung; Trenne die Schnittstellen von der Verarbeitungslogik.
- Das Verdrahten der Grundgedanken mit der Umsetzung hat zwei negative Effekte:
  - Es macht die Grundgedanken starr und schwierig den User Bedürfnissen anzupassen.
  - Das Ändern der Grundgedanken wird eine starke Tendenz haben die Umsetzung zu destabilisieren.

Die 17 Regeln aus „The Art of Unix Programming“

#5 Einfachheit

- Entwirf mit dem Ziel der Einfachheit; füge Komplexität nur hinzu, wenn es unbedingt sein muss.

Die 17 Regeln aus „The Art of Unix Programming“

#6 Sparsamkeit

- Schreibe nur dann ein großes Programm, wenn sich klar zeigen lässt, dass es anders nicht geht.

Die 17 Regeln aus „The Art of Unix Programming“

#7 Transparenz

- Entwirf mit dem Ziel der Durchschaubarkeit, um die Fehlersuche zu vereinfachen.
- Debugging kann sehr teuer werden, um diese Arbeitszeiten zu verkürzen kann man das Programm nach den Stichworten „Transparenz“ und „Feststellbarkeit“ entwickeln:
  - Transparenz: auf den ersten Blick ist klar was das Programm macht.
  - Feststellbarkeit: das Programm soll nicht nur gut funktionieren, sondern auch „von Aussen“ gesehen werden können, dass es funktioniert.

Die 17 Regeln aus „The Art of Unix Programming“

#8 Robustheit

- Robustheit ist das Kind von Transparenz und Einfachheit.
- Robustheit heisst, dass das Programm auch unter unerwarteten Konditionen genauso funktioniert wie unter normalen Konditionen.
- Man soll allgemein „Spezialfälle“ im Code vermeiden, dort entstehen die meisten Bugs.

Die 17 Regeln aus „The Art of Unix Programming“

#9 Darstellung

- Stecke das Wissen in die Datenstrukturen, so dass die Programmlogik dumm und robust sein kann.
- Komplexe Datenstrukturen sind, vergleichsweise zu einer Logik, einfach zu modellieren und zu begründen.

Die 17 Regeln aus „The Art of Unix Programming“

#10 kleinste Überrschungen

- Mache beim Entwurf der Schnittstellen immer das Nächstliegende, welches für die wenigsten Überraschungen beim Benutzer sorgt.

Die 17 Regeln aus „The Art of Unix Programming“

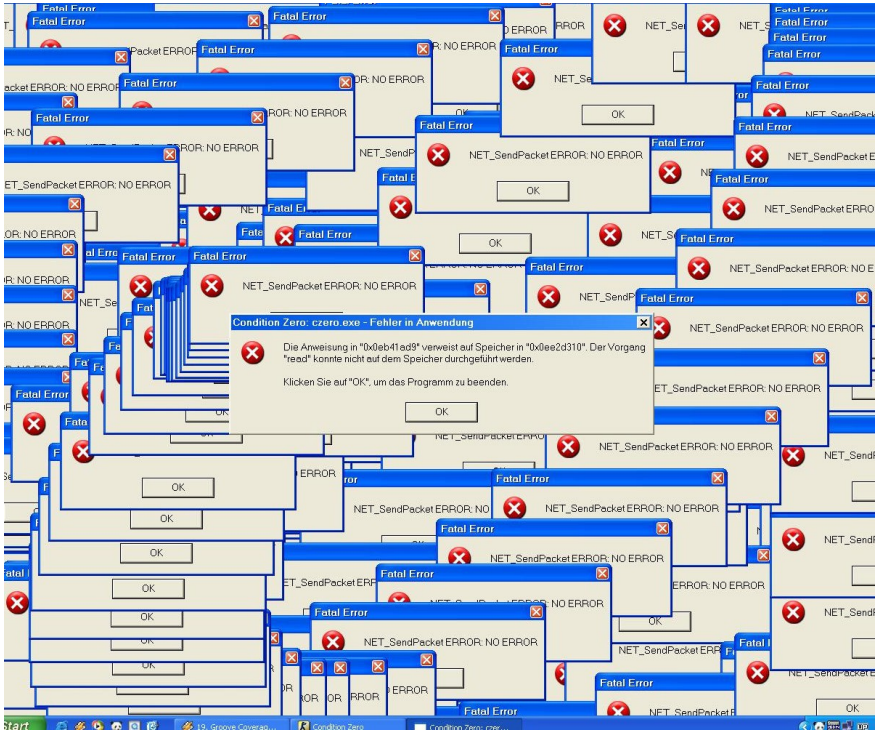
#11 Ruhe

- Wenn ein Programm nichts Überraschendes zu sagen hat, soll es schweigen.
- Wichtige Informationen sollen nicht mit gewöhnlichen Informationen vermischt werden.
- Die User-Aufmerksamkeit sollte als rares Gut gehandelt werden.

Die 17 Regeln aus „The Art of Unix Programming“

#12 Reperatur

- Wenn das Programm scheitert, soll es das lautstark und so früh wie möglich tun.
- Wenn das Programm eine Exception nicht behandeln kann, dann muss das Programm so zum „Absturz“ gebracht werden, dass man einfach Diagnosen stellen kann.



Die 17 Regeln aus „The Art of Unix Programming“

#13 Wirtschaftlichkeit

- Die Arbeitszeit von Programmierern ist teuer; spare sie auf Kosten der Rechenzeit.
- Maschinen Zyklen sind extrem günstig geworden.

Die 17 Regeln aus „The Art of Unix Programming“

#14 Generierung

- Vermeide Handarbeit; schreibe Programme, die Programme schreiben, falls möglich.
- Der Grund wieso wir Compiler und Interpreter entwickelt haben ist weil menschliche Handarbeit zu Verzögerungen und Fehlern führt.
- Beispiele: GUI Generatoren, Makefile Generatoren.

Die 17 Regeln aus „The Art of Unix Programming“

#15 Optimierung

- Erstelle Prototypen, bevor du dich an den Feinschliff machst. Mache es lauffähig, bevor du es optimierst.
- „Vorzeitige Optimierung ist die Wurzel alles Bösen“
- „Mach es zuerst lauffähig, bevor du es schneller werden lässt.“

Die 17 Regeln aus „The Art of Unix Programming“

#16 Vielfalt

- Misstraue allen Ansprüchen auf „den einzig wahren Weg“.
- Das Entwerfen unflexibler geschlossener Software ist eine ungesunde Form der Arroganz.

Die 17 Regeln aus „The Art of Unix Programming“

#17 Erweiterbarkeit

- Entwirf für die Zukunft, denn sie wird schneller kommen als du denkst.
- Beim Entwurf der Protokolle und der Dateiformate sollte darauf geachtet werden, dass sie selbsterklärend sind und erweiterbar.
- Lass Platz um die Datenformate und den Code wachsen zu lassen. Schaffe Schnittstellen um zukünftigen Programmierern ein einfaches anknüpfen zu ermögliche, ohne dass sie die ganze Architektur rekonstruieren müssen.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

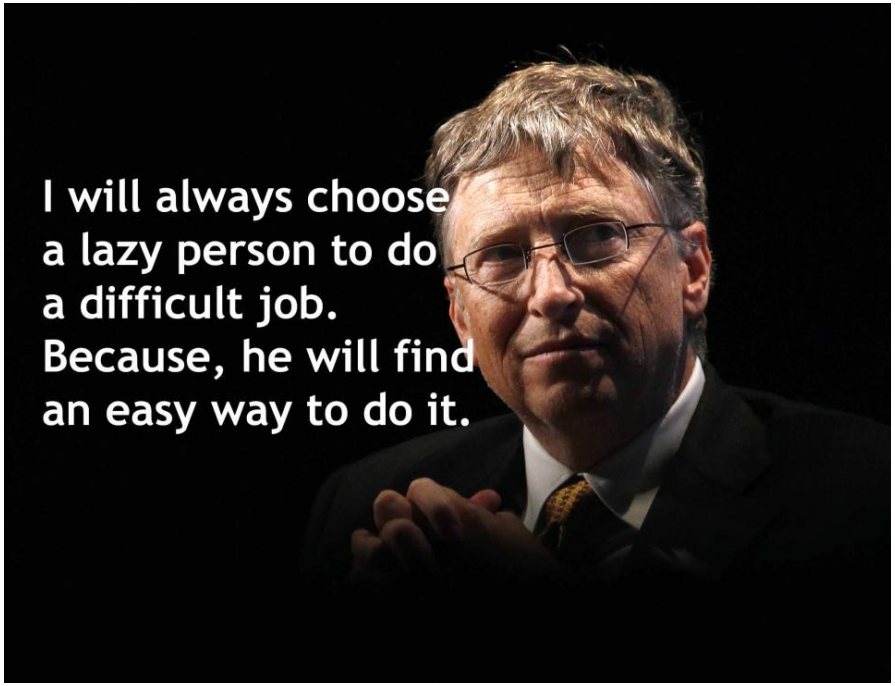
#1 „Every good work of software starts by scratching a developer’s personal itch.“

- Monotone, sich wiederholende Arbeitsschritte.
- Ich brauche etwas, aber ich glaube das kann ich besser

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#2 „Good programmers know what to write. Great ones know what to rewrite (and reuse).“

- Konstruktive Faulheit.
- Grundgedanke → Das Teilen des Codes in der Community.



Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#3 “Plan to throw one away; you will, anyhow.” (Fred Brooks, “The Mythical Man-Month”, Chapter 11)

- Im Nachhinein weiss man es immer besser.
- Die kritischen Probleme werden nach der ersten Implementation festgestellt.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#4 „If you have the right attitude, interesting problems will find you.“

- Beteilige dich an Open-Source Projekten
- Wenn du aktiv bleibst, wirst du mehr ins Projekt involviert.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#5 „When you lose interest in a program, your last duty to it is to hand it off to a competent successor.“

- Das Projekt würde verloren gehen.
- Suche einen kompetenten Nachfolger, der auch genügend Interesse mitbringt.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#6 „Treating your users as co-developers is your least-hassle route to rapid code improvement and effctive debugging.“

- Viele User können auch programmieren/haben ein Grundverständnis davon.
- Das Ermutigen der User Bugs zu melden bringt auch viele Vorteile:
  - Schon nur durch die grössere Menge an Testern werden mehr Bugs entdeckt
  - Jede Person nutzt ein Programm etwas anders.
  - User mit Programmierkenntnissen können das Problem diagnostizieren, oder Fixes vorschlagen.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#7 „Release early. Release often. And listen to your customers.“

- Je früher und schneller commited wird, desto schneller können Benutzer sich mit dem aktuellen Produkt beschäftigen und können Rückmeldungen geben.
- Es ist für Contributors ein Ansporn dran zu bleiben. Die Contributors sehen, dass ihre Arbeit veröffentlicht wird, eine kleine Belohnung um dran zu bleiben.

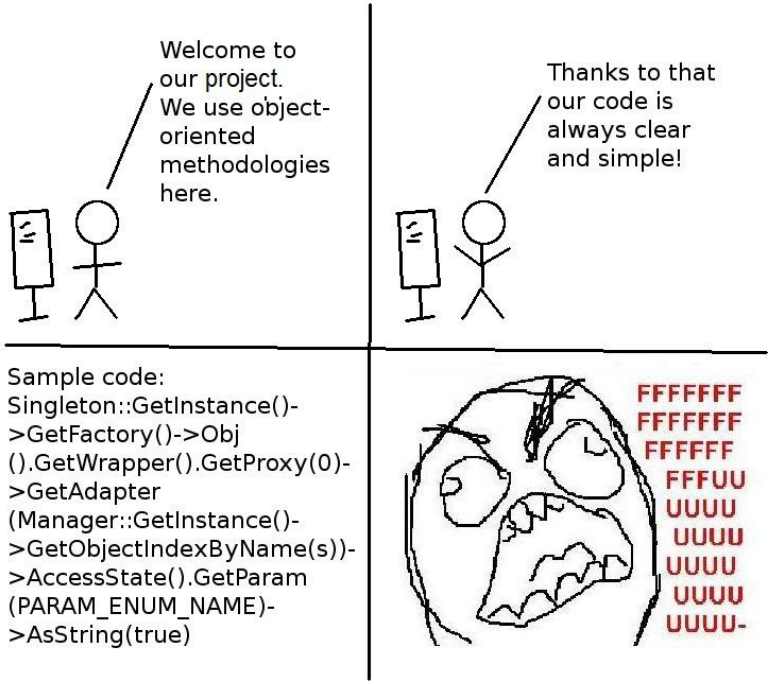
Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#8 „Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.“

- Hundert Augen sehen mehr als Zwei.
- In einer grösseren Menge an Menschen gibt es meistens einen der sich mit einem spezifischen Problem auskennt.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#9 „Smart data structures and dumb code works a lot better than the other way around.“



Die 19 Richtlinien aus „The Cathedral and the Bazaar"

#10 „If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource."

- „I got bug reports of a quality most developpers would kill for"
  - Viel frühe Realeases (in maximal 10 Tages Rhythmen)
  - Jeder aktive Userkontakt → Eintrag in die Beta-Mailingliste
  - Ermunterungen in den Mails mitzuhelfen.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#11 „The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.“

- Auch wenn man ehrlich ist und die Ideen, nicht als die Eigenen verkauft, wird dich die Community behandeln als wäre es eine Einzelleistung.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#12 „Often, the most striking and innovative solutions come from realizing that your concept of the problem was wrong.“

- Überdenken des Programmdesigns.
- Nicht zögern Überholtes aufzugeben.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#13 “Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away.”

- Code wird besser und einfacher → nahe am Ziel.
- Auch die Entwicklung und die Erforschung des Designraumes sind parallelisierbar.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#14 „Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.“

- Nach neuen Nutzungsmöglichkeiten Ausschau halten.
- Das Projekt sollte trotzdem simpel und robust bleiben.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#15 „When writing gateway software of any kind, take pains to disturb the data stream as little as possible – and *never* throw away information unless the recipient forces you to!“

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#16 „When your language is nowhere near Turing-complete, syntactic sugar can be your friend.“

- Kompaktheit ist heutzutage kein Selbstzweck mehr.
- Die Handhabung für den Programmierer sollte möglichst bequem sein.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#17 „A security system is only as secure as its secret. Beware of pseudo-secrets.“

- Man soll kein trügerisches Gefühl der Sicherheit vermitteln.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#18 „To solve an interesting problem, start by finding a problem that is interesting to you.“

- Probleme stellen sich als typisch dar für gewisse Klassen von Benutzern.

Die 19 Richtlinien aus „The Cathedral and the Bazaar“

#19 „Provided the development coordinator has a medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one.“

Truck Numbers

„How many or few would have to be hit by a truck (or quit) before the project is incapacitated?"

- Kann ermitteln ob es sogenannte „Heroes" in Projekten gibt. Heroes sind Personen die unverzichtbare Spezialisten sind für einen bestimmten Teil des Codes.
- FLOSS Projekte neigen dazu, sehr abhängig zu sein von solchen Heroes.
- Man sollte darauf achten, die Contributors möglichst gut auf den Code zu verteilen. Die Wahrscheinlichkeit eines Projektfehlschlags sinkt dadurch markant.
- Helfen kann dabei die Heroes einem komplett neuem/anderem Teil des Codes zuweisen, solange er noch verfügbar ist.

Quellenangabe

Lieratur:

- The Art of Unix Programming, Eric Steven Raymond (PDF: http://nakamotoinstitute.org/static/docs/taoup.pdf)
- The Cathedral and the Bazaar, Eric Steven Raymond (PDF: http://www.unterstein.net/su/docs/CathBaz.pdf)

Bilder:

Bild 1: http://www.codesimian.com/openSourceCommunism.jpg
Bild 2: http://thumbs.dreamstime.com/x/html-code-magnifier-glass-19126232.jpg
Bild 3: http://sd.keepcalm-o-matic.co.uk/i/keep-calm-and-printf-debugging.png
Bild 4: http://diginomica.com/wp-content/uploads/2014/07/power-of-silence.jpg
Bild 5: http://www.channelpronetwork.com/sites/default/files/thumbnails/slide-gallery//funny-error-messages.jpg
Bild 6: http://i.imgur.com/FFmFzxJ.jpg
Bild 7: http://photo3.ask.fm/309/185/300/710003016-1rnma2q-eri90p8j092h2f0/original/file.jpg
Bild 8: http://www.quickmeme.com/img/8e/8ef6afba7dcb0da2c0177572f832255b8f780c2a1c7c92faf15f11a58619ab11.jpg
Bild 9: http://www.dog-paw-print.com/image-files/dog-treat-recipes-3.jpg
Bild 10: http://www.nationalgeographic.de/thumbnails/lightbox/87/86/00/erdmaennchen-8687.jpg
Bild 11: http://www.asawicki.info/Download/Gallery/Misc/Ffuu/Ffuu_ObjectOriented.png
Bild 12: http://www.centercode.com/blog/wp-content/uploads/uncle-sam.gif
Bild 13: http://www.planet-wissen.de/natur_technik/insekten_und_spinnentiere/ameisen/img/ameisen_eichel_mauritius_g.jpg
Bild 14: http://www.dw.de/image/0,,3617591_4,00.jpg
Bild 15: http://www.dw.de/image/0,,5262144_4,00.jpg
Bild 16: http://i.dailymail.co.uk/i/pix/2013/10/15/article-2462025-18BFE80B00000578-475_634x389.jpg
Bild 17: http://wirsindeins.files.wordpress.com/2013/10/fragen-und-antworten-00.jpg

Vielen Dank für die Aufmerksamkeit!