Reimplementation in Gentoo

Institut für Informatik
FU Berlin
02.03.2009

Vortragsgliederung

Freie Universität Berlin

- Erster Artikel
  - Autor: Thomas Østerlie
  - „Distributed control in Gentoo Linux“
  - Vorgehensweise, Ablauf, Schlussfolgerungen
- Zweiter Artikel
  - Autoren: Thomas Østerlie und Letizia Jaccheri
  - „The Case of Changing a Large Open Source Software System“
  - Vorgehensweise, Ablauf, Analyse
- ~~Fazit~~
- Diskussion

Vorstellung der Autoren

Freie Universität Berlin

- Thomas Østerlie
  - 1973 in Trondheim geboren
  - Studierte an der Technisch-Naturwissenschaftlichen Universität Norwegens (NTNU)
  - 2003 Master of Science
  - Mittlerweile promoviert
  - Forscht am „Department of Computer and Information Science“ (IDI)

Vorstellung der Autoren

Freie Universität Berlin

- Thomas Østerlie (Fortsetzung)
  - Im Jahr 2000 IT-Berater für „Consult^{IT}“
    - Themenfelder Unix-Server und Computersicherheit
      - Unternehmen heißt Mittlerweile „Acando“
      - Momentanes Beschäftigungsverhältnis unbekannt
  - Schrieb Buchbesprechungen
    - z. B. zu „Hacking Linux Exposed“
  - Autor diverser wissenschaftlicher Artikel
    - Themen: Open Source, verteile Zusammenarbeit in großen Systemen
    - *„I study how software systems are developed; not how we ought to develop software […]“*
  - War/ist an der Entwicklung von einigen freien Programmen beteiligt

Vorstellung der Autoren

Freie Universität Berlin

- Letizia Jaccheri
  - 1965 in Pisa geboren
  - 1988 Master of Science
  - Arbeitete u. a. an der „Politecnico di Torino“
  - 1994 Promotion
  - Seit 1997 an der NTNU tätig
  - Seit 2002 Professorin für Software engineering am IDI
  - Inzwischen über 85 Artikel veröffentlicht, ca. 6 zusammen mit Thomas Østerlie

Erster Artikel

Freie Universität Berlin

- „In the network: Distributed control in Gentoo Linux“
  - Veröffentlicht 2004 in „Proceedings of the 4th Workshop on Open Source Software Engineering“
  - Empirische Studie über die Linux-Distribution *Gentoo*
  - Analysiert die Verteilung der Kontrolle im Projekt
    - Kontrolle hier: Die Macht ein Problem zu definieren und die angemessene Lösung festzulegen
    - Zentrale Kontrolle vs. verteilte Kontrolle

Kurzer Exkurs: Gentoo

- Gentoo Linux ist eine Linux-Distribution
  - Wird von der *Gentoo Foundation, Inc.* entwickelt
  - Richtet sich an fortgeschrittene Linux-Benutzer
  - Programme werden nicht als binäre Dateien sondern in Form des Quellcodes heruntergeladen
    - Benutzer muss diese selber kompilieren und konfigurieren
    - Dies dauert bei großen Programmen sehr lange
  - Vorteile: Individualität und maximale Konfigurierbarkeit
  - Keine großen Releases, sondern kontinuierliche Entwicklung
  - Dient als Grundlage für weitere Distributionen

Vorgehensweise

- Empirische Studie mit verschiedenen Datenquellen
  - Daten von der Gentoo-Website
  - Archive der Mailingliste
  - IRC Logs
  - Teilnahme am Projekt und gleichzeitige Beobachtung
  - Interview mit einem der Gentoo-Entwickler
- Artikel behandelt hauptsächlich ein Treffen der Hauptentwickler im Dezember 2003
  - Wöchentliches IRC-Treffen ist Teil der Organisationsstruktur des Gentoo-Projektes
  - (Zum damaligen Zeitpunkt noch alle zwei Wochen)
- Analyse der Situation mit Hilfe von Actor Network theory (Akteur-Netzwerk-Theorie, ANT)

Actor Network theory

Freie Universität Berlin

- Actor network theory
  - Gesellschaftswissenschaftliche Theorie zur Analyse der Beziehungen zwischen Akteuren
  - In den 80er Jahren entstanden
  - Besonders im englischsprachigen Raum verbreitet
  - Grundlage sind Netzwerke aus Knoten und Kanten
    - Akteure bilden Knoten
    - Assoziationen bilden Kanten
    - „Zusammenfassung der Interaktion zwischen den Akteuren“
  - Alle Entitäten sind Akteure, also auch Gegenstände
    - Symmetrie anstelle von Kategorien
    - Annahme, dass alle Entitäten aufeinander einwirken können
    - Soziologie öffnet sich für nichtmenschliche Akteure
    - „Kein Dualismus von Natur und Kultur“

Actor Network theory

Freie Universität Berlin

- Für Actor Network Analyse müssen Akteure und Assoziationen identifiziert werden
  - Akteure wollen ein Ziel erreichen
  - Assoziationen repräsentieren Verbindungen zwischen Akteuren
    - Netzwerk befindet sich im ständigen Wandel
    - Bildet nicht starr einen Zustand ab
  - Stabilität in einem Netzwerk kann abnehmen
    - Neue Akteure
    - Menschen ändern ihre Meinung
    - Maschinen gehen kaputt
  - Gesellschaft und Technik gemeinsam betrachten
    - Veränderung in einem bewirken eine Veränderung im anderen
- Beispiel: Wissenschaftler wollen ihre Reputation erhöhen
  - Dafür nutzen Sie Personen, Materialien, Techniken
  - Ziel: Verbündete mit selben Interessen finden, Macht erweitern

Actor Network theory

Freie Universität Berlin

- Akteure: Entitäten, die Dinge machen
  - Menschen, Maschinen, Geld, Nahrung, Bakterien, …
  - Ein Akteur will Interessen durchsetzen
  - Größe eines Akteurs hängt von der Größe des Netzwerkes ab, welches er kommandiert
    - Akteur will Verbündete mit gleichen Interessen gewinnen
    - Dadurch wächst Netzwerkgröße, also die Macht des Akteurs

- ANT ist nicht unumstritten
  - Kann Technik Interessen haben?
  - Können Dinge Macht ausüben?
  - Gleichheit von Mensch und Technik widersprüchlich

Ablauf der Studie

Freie Universität Berlin

- Portage ist zentraler Bestandteil von Gentoo
  - Verwaltet installierte Programme und kompiliert neue
- Mehrere Programme arbeiten direkt auf Konfigurationsdateien und Datenbanken von Portage
  - Das Format der Dateien hat sich mittlerweile geändert
  - Das führt zu Problemen durch veralteten Code
  - Neue API für Zugriff auf Konfigurationsdateien soll kommen
  - Zwei Entwickler werden dem Problem zugeordnet

Ablauf der Studie

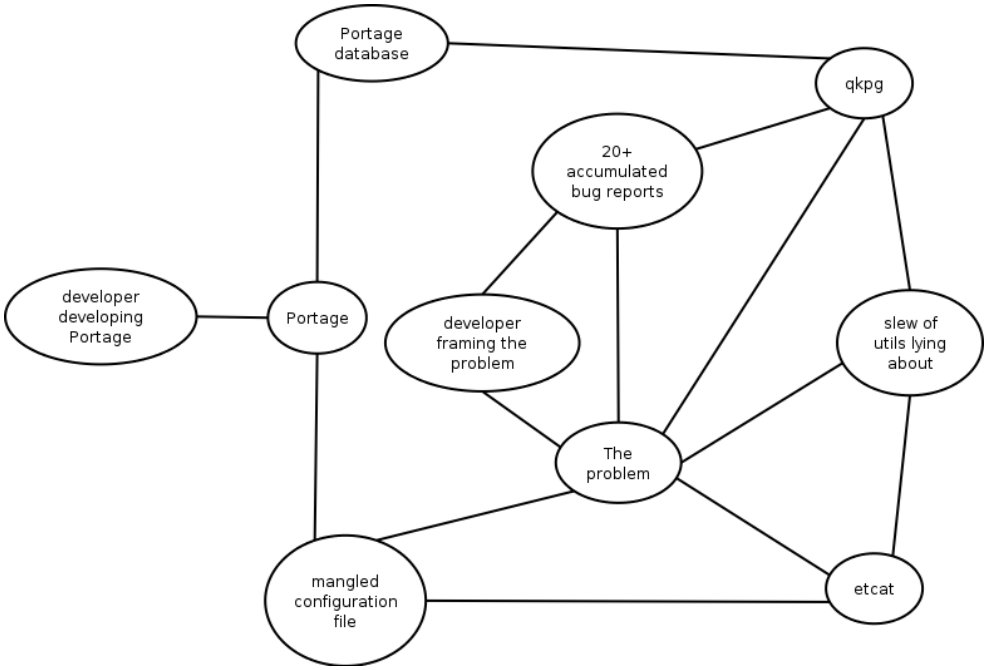
Freie Universität Berlin

- Chef-Entwickler: API basierend auf existierendem Portage-Code
- Entwickler stellen fest, dass für bestimmte Codeteile kein Maintainer existiert
  - Zuständige Entwickler haben das Projekt mittlerweile verlassen
- Es gibt mehrere Tools mit überlappenden Funktionen
  - Keines erledigt seine Arbeit wirklich gut
  - Jede Menge offene Bugs
  - Prioritätensetzung notwendig
  - *„We don't need five half-working use flag editors. We need one really good one“*
- Wer hat die Macht über Maßnahmen zu entscheiden?

„Framing the problem“

Freie Universität Berlin

- Problem wird von einem der Entwickler formuliert
- Als Antwort wird die Rolle des Maintainer eingebracht
  - Die Maintainer-Rolle nimmt an der Problemgestaltung teil
- Fehlerberichte werden Entwicklern zugewiesen
  - Solange Fehler ungelöst ist, wird der Entwickler erinnert
  - Dadurch rahmen auch Fehlerberichte das Problem mit ein



Schlussfolgerungen

Freie Universität Berlin

- Gentoo ist in Projekte und Unterprojekte aufgeteilt
  - Maintainer sind für jeweils einen Teil verantwortlich
  - Organisatorische Hierarchie
- ANT-Analyse offenbart Maintainer als Akteur im Netzwerk
  - Chef-Architekt kann seine Idee einer API nicht durchsetzen
  - Architekt in der Hierarchie über dem Entwickler
- Fazit: organisatorische Hierarchie und Kontrolle sind nicht unbedingt miteinander verbunden
  - Kontrolle stattdessen lokal im Netzwerk eingebunden
  - Definiert durch gegenseitige Beziehungen
  - Verteilt zwischen Akteuren
    - Verteilung nicht im geographischen Sinne
  - In diesem Fall zwischen Entwicklern, der Maintainer-Rolle und Fehlerberichten

Kritik zum Artikel

- Der Artikel ist teilweise schlecht formuliert
  - Beispiel: „By viewing of actors as inherently dispersed, thinking of the organization as an actor network shows that the hierarchical description of organization is just that: a hierarchical description of organization, an abstraction.“
- Abbildung wird nicht erklärt oder kommentiert
- Gedankengänge nicht immer nachvollziehbar
- Unklare Gesamtaussage des Artikels

Zweiter Artikel

Freie Universität Berlin

- „Balancing Technological and Community Interest: The Case of Changing a Large Open Source Software System“
  - Veröffentlicht 2007 in „Proceedings of the 30th Information Systems Research Conference (IRIS'30)“
  - Interpretierende Fallstudie über das Neuschreiben des Gentoo-Paketmanagers
    - Durch möglichst genaue Beobachtung und Beschreibung Aussagen über eine Sache machen
    - Ergebnisse sind nicht repräsentativ oder übertragbar

Kurzer Exkurs: Paketmanager

Freie Universität Berlin

- Ein Paketmanager verwaltet die installierte Software
  - Gehört zu praktisch jeder Linux-Distribution
  - Programme werden in Pakete verpackt und verteilt
  - Paketmanager ist die zentrale Stelle für Installation und Deinstallation von Programmen
  - Vorteile:
    - Abhängigkeiten werden automatisch aufgelöst
    - Installierte Programme können automatisch aktualisiert werden
    - Nicht mehr benötigte Pakete können einfach entfernt werden
  - Portage ist der Paketmanager von Gentoo Linux

Vorgehensweise

Freie Universität Berlin

- Datenquellen für die Fallstudie
  - Beobachtung und Analyse des IRC Channels über 10 Monate
  - IRC Logdateien der vorangehenden 11 Monate
  - Informelle Gespräche via E-Mail und IRC
  - Archive der Mailinglisten
  - Bugtracker
  - Versionsverwaltung
- Datenmaterial insgesamt von 2002 bis Ende 2005

Grundproblem

- Hauptinteresse der Endbenutzer ist Stabilität
- Das widerspricht größeren Änderungen
  - Je mehr Endbenutzer, desto schwieriger wird es
  - „inertia of the installed base“
- Änderungen erfordern daher eine Abwägung
  - Wie groß dürfen sie sein?
  - Wo und wann werden sie vorgenommen?

Ablauf

- Seit 2000 wuchs Gentoo zu einer der Top-Distributionen
- Existierender Paketmanager „Portage“ wird zur Belastung
  - Einige Anwendungen operieren direkt auf den Datenbanken und Konfigurationsdateien von Portage
  - *„The source code is very fragile because it has evolved rather than being designed“*
  - Alle Entwickler erkennen die Notwendigkeit den Paketmanager zu ersetzen
  - Nur wenige sind dazu in der Lage

Ablauf

- Erster Versuch im November 2003
  - Vier Entwickler machen Ankündigung für „Portage-ng“
  - Modularer Ansatz, stabile API
  - Komponenten sollen von der Community kommen
  - Prototyp wird zwecks Robustheit in Prolog entwickelt
    - Programmiersprache weckt Widerstand in der Community
    - Bedenken bezüglich der Geschwindigkeit
    - Nur wenige beherrschen Prolog
  - Im Dezember 2003 erscheint konkurrierender Prototyp
    - Geschrieben in Ada
    - Endlose Diskussionen über richtige Programmiersprache
  - Bis Februar 2004 kommt Portage-ng zum Stillstand

Ablauf

- Zweiter Versuch Februar 2004
  - Modularisierung des vorhandenen Systems zu „Portage-mod“
  - Parallele Änderungen am existierenden Portage machen Ansatz zunichte
- Dritter Versuch
  - Idee: API über die vorhandenen Anwendung schreiben, als Fortsetzung des zweiten Versuches
    - Weniger Abhängigkeiten
    - Änderungen am Format der Konfigurationsdateien möglich
  - Nach 1,5 Monaten wird der Versuch abgebrochen
    - Erweist sich als ungeeignet für Fernwartung
    - Neuer Ansatz: Unix Daemon

Ablauf

- Bis November 2006 nur Bugfixes und kleinen Änderungen
  - Alle Entwickler sind sich einig, dass der existierende Paketmanager ersetzt werden muss
  - Alle Versuche scheitern über mehrere Jahre

Analyse

Freie Universität Berlin

- Anwendungen arbeiten direkt auf Konfigurationsdateien
- Nur vier Entwickler sind zu Änderungen in der Lage
- Wartung ist parallel zur Neuentwicklung erforderlich
  - Manche Entwickler sehen Rewrite als Ressourcenverschwendung
  - Rewrite nur möglich, wenn existierendes Portage für 6-12 Monate ohne Updates bleiben kann
- Portage-ng scheitert, da keine Ressourcen für die Entwicklung der Module mobilisiert werden können
  - Viele Ressourcen werden in Prolog-Diskussionen gebunden
  - Kern-Entwicklung hat nicht genügend Entwickler
  - Prototyp wird niemals fertig
  - Idee des „Kondensationskeims“ scheitert

Analyse

- Konkurrenzkampf innerhalb von Gentoo
- Es fehlt klare Übergangsstrategie
  - Ständige Diskussionen über Umfang und Reihenfolge der Änderungen
  - Fähigkeiten des existierenden Systems zählen stärker als die Vorteile, die ein Systemwechsel hätte

- Wie hätte man es besser machen können?

Diskussion!