Praktikum angewandte Systemsoftwaretechnik

Aufgabe 4

Alexander Würstlein, Moritz Strübe, Rainer Müller

Lehrstuhl Informatik 4

Mai 2013

Linux-Upstream-Entwicklung

- „*The Cathedral and the Bazaar*" (Eric S. Raymond)
  - Essay über Methoden der Software-Entwicklung
  - Basiert auf Beobachtungen des Entwicklungsprozesses des Linux-Kern

- Hauptaussagen

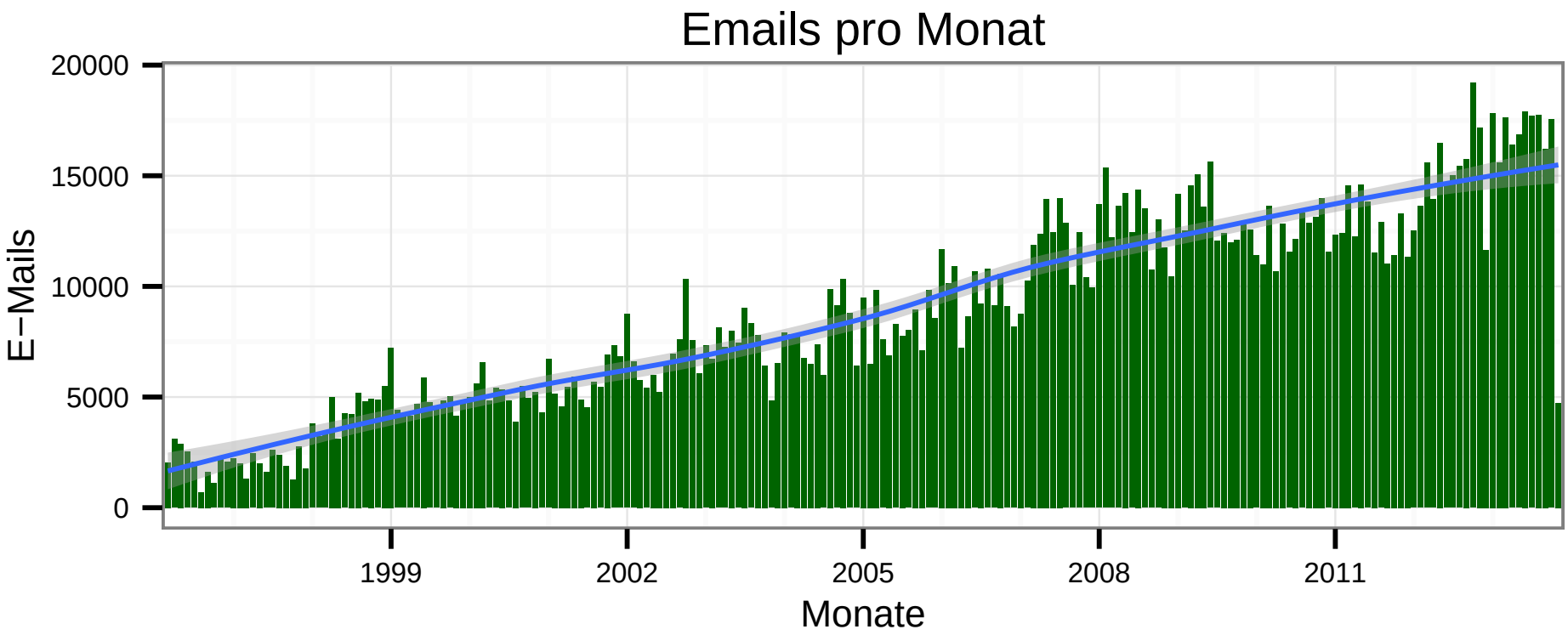
  - „Every good work of software starts by scratching a developer's personal itch."
    Kapitel 2: „*The Mail Must Get Through*"

  - „Release early. Release often"
    Kapitel 4: „Release Early, Release Often"

  - „If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource."
    Kapitel 5: „*Is A Rose Not A Rose?*"

Linux-Upstream-Entwicklung (Fortsetzung)

- Gesamte **relevante** Kommunikation ausschließlich via E-Mail
- Wichtigster Kommunikationskanal: Linux-Kernel-Mailing-List (LKML)



- Derzeit ca. 7400 Abonnenten, Verbreitung auch über RSS-Feeds, News u.Ä.

Programmcode-Integration

- Source-Code des Linux-Kern
  - Jede Veränderung am Linux-Kern durchläuft den gleichen Prozess
  - Egal ob Fehlerbehebung oder neue Funktionalität
- Kreuzgutachten („*Peer-Review*")
  - Gewachsene Hierarchie mit „*Benevolent Dictator For Life*"
  - *Meistens* siegt der überlegene Ansatz
- Einflussreichste Aktoren

Linus Torvalds

Andrew Morton

Alan Cox

Greg Kroah-Hartman

Kernel Janitors - Die Hausmeisterei des Linux-Kerns

- Im wesentlichen eine Gruppe von Kernel Entwicklern die kleinere, weniger gravierende Patches entgegen nimmt, sammelt, bewertet und gebündelt in den Entwicklungsprozess einbindet.
- Unterprojekt von `http://kernelnewbies.org`
- Hervorragende, leider leicht angestaubte Anlaufstelle für Neulinge im Linux Entwicklerumfeld

- Homepage: `http://kernelnewbies.org/KernelJanitors/`

Einsenden von Entwicklungen in den Linux-Kern

Nützliche Hinweise zum Prozess in `Documentation/SubmittingPatches`

- Patches im *unified diff* Format
- Sinnvolle und nachvollziehbare Beschreibungen der Änderungen
- Nur **eine** logische Änderung pro Patch
- Richtige(n) Addressat(en) finden
- Keine Dateianhänge, kein MIME, kein HTML
- Basisversion genau angeben
- Knappe (70-75 Zeichen!) Kurzbeschreibung, darunter dann ausführliche Beschreibung des Patches und dessen Intention.
- Unnötige Diskussionen über Geschmack (etc.) erkennen, richtig begegnen und vermeiden ($\rightsquigarrow$ siehe auch: `http://bikeshed.org`)

Andi Kleen: „On submitting kernel patches"
`http://halobates.de/on-submitting-patches.pdf`

Auffinden von zuständigen Betreuern

- Änderungen an Dateien im Linux-Kern sind immer an den jeweiligen Betreuer zur Begutachtung einzusenden
- Aufgrund der schieren Größe ist die Zuständigkeit der jeweiligen Dateien aufgeteilt
- Hilfsmittel: `scripts/get_maintainer.pl`:

```
$ scripts/get_maintainer.pl -f fs/btrfs/volumes.c
scripts/get_maintainer.pl -f fs/btrfs/volumes.c
Chris Mason <chris.mason@oracle.com>
linux-btrfs@vger.kernel.org
linux-kernel@vger.kernel.org
```

- Kann auch auf einen Patch direkt angewendet werden, um direkt die betroffenen Dateien und deren Betreuer zu finden.

Linux Coding Style

- Quelltext in Linux werden *normalisiert* bearbeitet
- `http://lxr.linux.no/linux/Documentation/CodingStyle`
- Im Einzelnen: Einrückung, Lange Zeilen, Setzen von Klammern und nichtdruckbare Zeichen (Whitespace), Bezeichnernamen, Tabulatoren, Kconfig, etc.
- automatisierter Test: `scripts/checkpatch.pl`

$\Rightarrow$ Erleichtert das Lesen und Verständnis, vermeidet Auseinandersetzungen

$\Rightarrow$ Verstöße gegen die Richtlinien führen häufig zur Ablehnung des Patches

Entwicklungen absegnen

- In der Linux-Entwicklung wird großer Wert auf korrekte Zuordnung von Patches zu Maintainern gelegt
- Bei einzelnen, kleineren Patches ist häufig unklar, wer der eigentliche Autor ist und was die genauen Nutzungsbedingungen sind
- Per Konvention werden daher Patches von ihren jeweiligen Autoren mit einer speziellen Notation in der Commit-Nachricht *abgesegnet:*

```
Signed-off-by: Random J Developer <random@developer.example.org>
```

Bedeutung (Auszug, Details in `Documentation/SubmittingPatches`):

- Der Autor bestätigt das Werk ganz oder in Teilen selbst geschrieben zu haben.
- Der Autor bestätigt das Recht zur Veröffentlichung zu haben
- Der Autor erlaubt die Verwendung und den Vertrieb der Änderung unter den Bedinungen der ursprünglichen Version

Weitere Absegnungen

- `Reported-by:` Wer hat das Problem (richtig) gemeldet
- `Reviewed-by:` Wer hat die Änderung begutachtet
- `Tested-by:` Wer hat den Patch getestet
- `Acked-by:` Patch ist zur Kenntnis genommen worden, nicht notwendigerweise aber getestet

Beispiel für Absegnungen

```
Subject: [PATCH] ACPICA: Fix possible fault in return package
object repair code

Fixes a problem that can occur when a lone package object is
wrapped with an outer package object in order to conform to
the ACPI specification. Can affect these predefined names:
_ALR,_MLS,_PSS,_TRT,_TSS,_PRT,_HPX,_DLM,_CSD,_PSD,_TSD

https://bugzilla.kernel.org/show_bug.cgi?id=44171

The bug got introduce by commit 6a99b1c94d053b3420eaa4a4bc
in v3.4-rc6, thus it needs to get pushed into 3.4 stable
kernels as well.

Reported-by: Vlastimil Babka <caster@gentoo.org>
Tested-by: Vlastimil Babka <caster@gentoo.org>
Tested-by: marc.collin@laboiteaprog.com
Signed-off-by: Bob Moore <robert.moore@intel.com>
Signed-off-by: Lin Ming <ming.m.lin@intel.com>
CC: stable@vger.kernel.org
---
 drivers/acpi/acpica/nspredef.c |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)
```

„Schöne“ Patchserien bauen mit git

- `git commit --amend`: die vorherige Commit-Nachricht ändern
  - kann ggf. mit `git add` markierte Änderungen hinzufügen
  - z.B. auch mit `--reset-author` oder `--author ...`
- `git checkout abcde -b foo`: einen neuen Seitenzweig „foo“ bei commit abcde auschecken
- `git cherry-pick 12345`: nur Commit „12345“ in aktuellen Zweig übernehmen
- `git rebase -i`: interaktiv eine Serie von Commits editieren
  - ausführliche Beschreibung: `http://git-scm.com/book/en/Git-Tools-Rewriting-History`
- `git add -p`: nur Teile einer Datei für einen Commit auswählen

Übungsaufgabe #4

- Programmcode des Linux-Staging-Bereich des aktuellen Staging-Baums analysieren und Defekte finden
- Patches für gefundene Fehler erstellen und ausreichend testen:
  - Patch auf Sourcen des Linux-Kernel anwenden
  - Kompilieren und sicherstellen, dass der betroffene Code übersetzt wurde
  - Überprüfung des Patches mit `scripts/checkpatch.pl`
- Patches an linux-kernel@i4.cs.fau.de senden
- Einsendung der Patches an `devel@linuxdriverproject.org` mit Kopie (`CC:`) an die zuständigen Maintainer (`get_maintainer.pl`) und linux-kernel@i4.cs.fau.de
- Vorstellung der Ergebnisse in der Tafelübung (Kurzvortrag, maximal drei Folien pro Gruppe)

Vorstellung der Ergebnisse in der Tafelübung am 2014-01-20