Software-Entwicklung mit Linux

Matthias Kalle Dalheimer

O'Reilly Verlag, Köln/Hamburg

22.05.1997

- Was ist Software-Entwicklung eigentlich?
- Pinguine im Turm zu Babel oder: Der Sprachenzoo
- GUI-Toolkits
- Datenbanken
- Organisation größerer Programmierprojekte

Was ist Software-Entwicklung eigentlich? I

- Mehr als

  ```
  vi hello.c ; gcc -o hello hello.c ; ./hello ; \
  gdb hello
  ```

- Erfassen des Problems und der Problemdomäne
- Zusammenarbeiten mit den späteren Benutzern
- Auswählen der richtigen Werkzeuge
- Entwicklung planen

Was ist Software-Entwicklung eigentlich? II

- Entwickeln und Dokumentieren
- Selber testen
- Von den Benutzern testen lassen
- Archivieren
- Wiederverwenden?

Pinguine im Turm zu Babel I

Ada, APL, Basic, C, C++, Objective-C, Eiffel, Forth, Fortran, Guile, Java, JavaScript, Lisp, Logo, ML, Modula-2, Modula-3, Oberon, Pascal, Perl, Prolog, Python, Rexx, Sather, Scheme, Simula, Smalltalk, Tcl, ...

Pinguine im Turm zu Babel II — Eiffel

- Kreuzung aus C++, Oberon und Smalltalk
- Ziele: Wiederverwendbarkeit, Erweiterbarkeit, Verläßlichkeit, Effizienz, Offenheit, Portabilität
- Compiler übersetzen meist in C-Code
- Ca. drei Compiler für Linux verfügbar
- rein objekt-orientiert (z.B. keine „freischwebenden“ Funktionen)
- Garbage Collection
- stark getypt

Pinguine im Turm zu Babel III — Eiffel (Fortsetzung)

- Klassen haben bestimmte Mengen von „Features“, die an genau definierte andere Klassen exportiert werden.
- Objekte werden normalerweise (wie in Java) als Referenzen angesprochen.
- „Konstruktoren“(*creation procedures*) sind ganz normale Prozeduren, die speziell markiert werden.

Pinguine im Turm zu Babel IV — Eiffel (Fortsetzung)

- Wichtiges Sprachmittel: Zusicherungen (*Assertions*)
  - Ermöglicht *Design by contract*
  - Vorbedingungen (*preconditions*)
  - Nachbedingungen (*postconditions*)
  - Invarianten (*invariants*)
- *Short Form* als semi-automatische Klassendokumentation

Pinguine im Turm zu Babel V — Eiffel (Fortsetzung)

- Exception Handling mit „zweiter Chance“
- Formale generische Parameter
- (Mehrfach-)Vererbung, Polymorphie, dynamische Bindung, Aliasing
- „Verzögerte“ Klassen/Routinen

Pinguine im Turm zu Babel VI — Prolog

- Logisches Programmieren
- Prädikatenlogik (Horn-Klauseln)
- Typische Einsatzgebiete: Expertensysteme, maschinelle Sprachverarbeitung (Machine Translation, Zugangssysteme, textverstehende Systeme, . . .)
- Nicht lösungsorientiert, sondern problemorientiert arbeiten.

Pinguine im Turm zu Babel VII — Prolog (Fortsetzung)

- Programm besteht aus
  - Problemabhängiger Wissensbasis (Fakten und Regeln)
  - Problemunabhängiger Inferenzmaschine (vom System bereitgestellt)
- Der Ablauf eines Prolog-Programms ist das Beweisen eines Theorems.
- Verfahren: Tiefensuche mit Backtracking (andere Suchverfahren können mit wenig Aufwand selbst implementiert werden)

Pinguine im Turm zu Babel VIII — Prolog (Fortsetzung)

- Häufig ist man aber eher an den Nebeneffekten (den Instantiierungen der Variablen) als an beweisbar/nicht beweisbar interessiert.
- Vom System-bereitgestellte extra-linguistische Prädikate sorgen für Ein-/Ausgabe, Bildschirmsteuerung, etc.
- Für Linux: SWI-Prolog (einschließlich GUI-Toolkit XPCE)

Pinguine im Turm zu Babel IX — Prolog (Fortsetzung)

Formulierung von Fakten:

```
spricht(linus_torvalds, finnisch).
spricht(linus_torvalds, schwedisch).
spricht(linus_torvalds, englisch).
spricht(kalle_dalheimer, deutsch).
spricht(kalle_dalheimer, schwedisch).
spricht(dirk_hohndel, deutsch).
spricht(dirk_hohndel, englisch).
mag(linus_torvalds, pinguine).
mag(kalle_dalheimer, tiger).
mag(dirk_hohndel, baeren).
```

Pinguine im Turm zu Babel X — Prolog (Fortsetzung)

Anfragen:

- Spricht Linus Torvalds englisch?

```
?- spricht(linus_torvalds, english).
Yes
```

- Welche Sprachen spricht Dirk Hohndel?

```
?- spricht(dirk_hohndel, X).
X = deutsch ;
X = englisch ;
No
```

- Können sich Kalle Dalheimer und Linus Torvalds in einer Sprache unterhalten?

```
?- spricht(kalle_dalheimer, X),
      spricht(linus_torvalds, X).
X = schwedisch ;
No
```

Pinguine im Turm zu Babel XI — Prolog (Fortsetzung)

- Regeln: Jemand der schwedisch spricht, mag auch Bären.

```
mag(X, baeren) :-
     spricht(X, schwedisch).
```

- Neue Anfrage:

```
?- mag(X, baeren).
X = linus_torvalds ;
X = kalle_dalheimer ;
X = dirk_hohndel ;
No
```

GUI-Toolkits I — Klassifikation

- Text-basiert vs. X basiert
- C vs. C++ (und andere?)
- nur Unix vs. plattformübergreifend (OS/2, Macintosh, ...)
- zu Bildungszwecken geeignet vs. für Großprojekte geeignet

GUI-Toolkits II — Text-basierte

- curses/ncurses
  - Klassiker der Bildschirmansteuerung
  - Farbe
  - Menüs
  - Fenster mit Backing Store
- Slang

GUI-Toolkits III — Motif

- Industrie-Standard
- kommerziell
- Sehr viel Literatur erhältlich
- Umständlich zu programmieren
- Eigentlich nicht als Toolkit, sondern als Grundlage dafür gedacht
- Langsam, viele Fehler
- C, aber einige C++-Wrapper verfügbar
- Unix/X11

GUI-Toolkits IV — wxWindows

- frei (Berkeley-ähnliche Lizenz)
- Unix/X11, MS-Windows (Macintosh und OS/2 in Arbeit/Planung)
- basiert auf Motif, Athena oder XView
- C++
- sehr aktive Benutzerbasis
- viele zusätzliche Widgets erhältlich
- Aussehen der Programme entspricht der verwendeten Plattform
- GUI-Designer wxBuild

GUI-Toolkits V — Qt

- kommerzielles Produkt, aber frei für freie Software
- siehe Vortrag von Haavard Nord
- C++ (Python-Bindung in Arbeit)
- Unix/X11 und MS-Windows
- basiert unter Unix direkt auf Xlib, Motif-Look-and-Feel
- hohe Qualität der Bibliothek und der Dokumentation
- guter Support, auch für die freie Lizenz
- Basis für das KDE-Projekt (siehe Vortrag von Matthias Ettrich)

GUI-Toolkits VI — V

- frei
- C++
- basiert unter Unix auf Athena-Widgets
- für Unterrichtszwecke entworfen, aber auch für kleinere Projekte geeignet
- Dialoge, Menüs etc. werden in statischen Strukturen abgelegt

GUI-Toolkits VII — XForms

- frei für nicht-kommerzielle Projekte, Source-Code nicht verfügbar
- ungewöhnliches Look-and-Feel
- Programme sehr schnell
- guter Dialog-Builder Bestandteil
- C, nur Unix

Datenbanken I — Klassifikation

- Relational vs. objekt-relational vs. objekt-orientiert vs. hierarchisch
- Zugriffsmöglichkeiten
  - SQL-Monitor
  - Embedded SQL
  - C-, C++-, Perl-, Python-, Tcl-API
  - Grafische Tools
  - ...
- Referentielle Integrität unterstützt?
- Client/Server vs. lokale Dateien

Datenbanken II

- Postgres
  - frei, objekt-relational
  - SQL-Monitor, C-, C++-, Perl-, Python-, Tcl-API
  - Client/Server
- Yard SQL
  - kommerziell, Private Edition für Linux erhältlich
  - relational
  - Unterstützt Referentielle Integrität
  - SQl-Monitor, grafische Schnittstelle, ESQL
  - Client/Server und lokal

Datenbanken III

- Justlogic SQL
  - kommerziell, Private Edition für Linux erhältlich
  - relational, lokale Dateien
  - ESQL, C-, C++-API
- mSQL, MySQL
  - kommerziell/frei
  - relational, Client/Server
  - C-, C++-, Perl-API
  - Basis für viele Web-Datenbanken

Datenbanken IV

- FlagShip
  - Clipper-Clone, DOS/MS-Windows xBase-Programme können oft ohne Anpassung übernommen werden
  - kommerziell, Test Drive für Linux verfügbar
- Qddb
  - nur für kleinere Projekte geeignet (eigentlich keine Datenbank im klasischen Sinne)
  - Tcl-API, sehr schönes Toolkit erzwingt einheitliches Look-and-Feel aller Qddb-Datenbanken
  - Bspw. Literaturdatenbank (kommerziell) erhältlich

Organisation größerer Programmierprojekte I

Probleme:

- Wer macht was?
- Wer macht was wann?
- Koordination in „Projekten ohne Projektleiter“
- Release-Management

Organisation größerer Programmierprojekte II — CVS

- Versionskontrolle für mehrere Verzeichnisse
- Kein Locking nötig, CVS unterstützt beim Zusammenführen der Stände mehrerer Entwickler
- Aktualisierung auch per Client/Server und per Mail möglich

Organisation größerer Programmierprojekte III — GNATS

- Datenbank mit Problem Reports (PRs)
- PRs werden in vorgegebenem Format per Mail geschickt.
- Funktionen zum Anlegen und Editieren von PRs und zum Auswerten der Datenbank
- Installationsaufwand eher gering, aber relativ hoher organisatorischer Aufwand, erfordert daher gutes Projektmanagement.

Weitere Informationen

- `ftp://sunsite.unc.edu/pub/Linux/devel`
- Linux Applications and Utilities Page (`http://www.chariott.com/linapps.html`)
- Linux Buyer's Guide
- `<self-promotion type=shameless>`

  Matthias Kalle Dalheimer:
  Software-Entwicklung mit Linux (Arbeitstitel)
  O'Reilly Verlag
  erscheint voraussichtlich August 1997

  `</self-promotion>`