

Architektur Distributions-Management am Beispiel Debian

Jan Kechel
Martin Klink

Architekturtheorie
Sommersemester 2006
Prof. Dr. Bernd Mahr
Technische Universität Berlin

Abstract

In der Lehrveranstaltung Architekturtheorie haben wir viele unterschiedliche Ansätze behandelt Architekturen zu beschreiben. Diese Ausarbeitung versucht die vermittelten Methoden zur Architekturbeschreibung auf Distributions-Management, wie es heutzutage bei vielen gängigen Linux-Distributionen zum Einsatz kommt, anzuwenden. Als konkretes Beispiel beschreiben wir das System des Distributors Debian [4].

1. Einleitung

Für die Beschreibung der Architektur orientieren wir uns an den Viewpoints des ISO/IEC Standards "Open Distributed Processing - Reference Model" (RM-ODP)[16]. Allerdings ist darauf hinzuweisen, daß diese Beschreibung in keinster Weise den Anspruch erhebt RM-ODP konform zu sein. Vielmehr nutzen wir die von RM-ODP beschriebene Strukturierung, und erhoffen so eine übersichtliche Einführung in das System geben zu können.

2. Begriffe

Zum besseren Verständniß der folgenden Architekturanalyse möchten wir zu Anfang drei Begriffe definieren:

Distributor Eine Community die Aktivitäten durchführt, um Software an Software-Benutzer zu verteilen.

Distributions-Management Das System, welches der Distributor benutzt, um seine Ziele zu erreichen.

Distribution Die Menge aller Software, die einem Benutzer über das Distributions-Management eines Distributors bereitgestellt wird.

3. Enterprise Viewpoint

3.1. Purpose

Wenn man sich mit den Zielen des Distributors Debian beschäftigt, erkennt man sehr schnell, dass es sich nicht um eine Firma mit wirtschaftlichen Interessen handelt. Ein Satz aus dem Debian Policy Manual [9] besagt: "The Debian Project is an association of individuals who have made common cause to create a free operating system." Bei Debian handelt es sich also um eine Gruppe von Menschen, die ein gemeinsames Ziel haben. Dieses ist grob gesagt, ein freies Betriebssystem bereitzustellen, wobei Debian eine genaue Definition von "frei" im Kapitel 2.1 der Debian Policy [9] liefert.

Da eine möglichst grosse Breite an Programmen angeboten werden soll, wird über das Distributionsmanagement von Debian auch Software verwaltet, die zwar kostenlos aber nicht "frei" verfügbar ist. In solchen Fällen werden Benutzer aber darauf hingewiesen und die entsprechende Lizenz wird mitgeliefert. Solche Programme sind allerdings nicht teil der eigentlichen Debian-Distribution.

Ein weiteres Ziel, daß sich aus der Bereitstellung von kompletten Systemen ergibt, ist Automatisierung. Distributionsmanagementsysteme versuchen deshalb Tools bereitzustellen, die es dem Nutzer möglichst einfach machen Software in ihr System zu integrieren, wieder zu entfernen und zu konfigurieren.

Der Distributor Debian verfolgt noch andere spezielle Ziele. Zum einen soll jeder dazu gebracht werden freie Software zu schreiben und es soll auch jeder im Stande sein die Debian Distribution zu verteilen ohne dabei Lizenzen, Einschränkungen oder Gesetze zu brechen.

3.2 Scope

Der Anwendungsbereich von Distributionsmanagementsystemen erstreckt sich grundsätzlich über zwei

Gebiete. Zum einen die Distributionserstellung und zum anderen die Distributionsverteilung. Vor allem bei der Distributionserstellung wird dabei noch viel händisch erledigt. Die Automatisierung ist hierbei oftmals nur Unterstützend.

Die Verteilung der Distribution ist dagegen ein weitgehend automatisiertes System, welches nur in Fehlerfällen oder für individuelle Konfigurationen Eingriffe der Benutzer erfordert.

3.2.1 Rolle Distributions-Ersteller

Der Distributions-Ersteller hat die Aufgabe eine Distribution entsprechend verschiedener Vorgaben zusammenzustellen. Das wohl wichtigste Merkmal das die Programme der Distribution erfüllen müssen sind die Lizenzbestimmungen. Debian hat dazu eine eigene Liste von Kriterien für freie Software, die "*Debian Free Software Guidelines*" [7]:

Freie Weiterverteilung Die Lizenz einer in die Distribution aufgenommenen Komponente darf niemandem verbieten diese Komponente selber weiterzugeben oder zu verkaufen. Die Lizenz darf keine Gebühren für einen Weiterverkauf verlangen.

Source Code Jedes Programm muss den Sourcecode enthalten, sowie die Verteilung muss sowohl in Source- als auch in Binär-Form erlaubt sein.

Abgeleitete Werke Die Lizenz muss Veränderungen und abgeleitete Werke gestatten, außerdem muss die Weiterverteilung dieser veränderten Werke unter der selben Lizenz ebenfalls gestattet sein.

Unversehrtheit des Sourcecodes Falls eine Lizenz die Verteilung des Source-Codes in veränderter Form nicht gestattet, so muss aber die Verteilung in Kombination mit patch Dateien erlaubt sein. Die Verteilung veränderter Programme in binärform muss in jedem Fall gestattet sein.

Keine Diskriminierung gegen Personen Die Lizenz darf keine Personen oder Gruppierungen diskriminieren.

Keine Einschränkung des Anwendungsgebiets Die Lizenz darf die Anwendung der Software nicht auf bestimmte Einsatzgebiete beschränken.

Verteilung der Lizenz Alle Rechte an dem Programm müssen für alle gelten an die das Programm verteilt wird. Es dürfen keine zusätzlichen Regelungen in Kraft treten.

Die Lizenz darf nicht spezifisch für Debian sein

Alle Rechte die die Lizenz gewährt dürfen nicht auf Debian Systeme beschränkt sein. Wenn das Programm aus der Debian Distribution herausgenommen und ohne Debian entweder benutzt oder weiterverteilt wird müssen ebenfalls genau die gleichen Rechte gelten wie innerhalb des Debian Systems.

Keine Seiteneffekte auf andere Programme

Es dürfen keine Beschränkungen auf andere Programme enthalten sein. Zum Beispiel darf eine Lizenz nicht verlangen daß ein Programm nur mit anderer, freier Software, zusammen verteilt werden darf.

Beispiel Lizenzen die Debian als frei bezeichnet sind zum Beispiel die GPL [14], BSD [3], und die Artistic Lizenz [2].

Da es aber leider viele Programme gibt, die diesen Richtlinien von frei nicht genügen, aber trotzdem von vielen Endanwendern entweder benötigt oder gerne benutzt werden, gibt es eine Kennzeichnungspflicht für solche Programme, die trotz Verletzung der DFSG mit in die Distribution aufgenommen werden sollen. Es wird unterschieden zwischen *main*, *contrib* und *non-free*.

Mit der Kennzeichnung *main* dürfen nur Programme versehen werden die alle Vorgaben der DFSG erfüllen und nur von Programmen abhängig sind die selber mit *main* gekennzeichnet sind. Programme die zu viele Fehler enthalten oder eine der Debian policy requirements verletzen dürfen trotzdem nicht mit *main* gekennzeichnet werden.

Mit der Kennzeichnung *contrib* müssen Programme versehen werden, die alle Kriterien für eine Kennzeichnung mit *main* erfüllen, bis auf die Abhängigkeiten. In *contrib* dürfen also Programme enthalten sein die zum Beispiel von Programmen mit der Kennzeichnung *contrib* oder *non-free* abhängen.

Mit der Kennzeichnung *non-free* werden alle Programme versehen, die nicht den DFSG entsprechen.

Zur eigentlichen Aufnahme eines Programms in die Distribution muss ein klar definiertes Dateiformat (Debian Paket) eingehalten werden, in dem neben dem eigentlichen Programm auch die Lizenz sowie Abhängigkeiten zu anderen Programmen enthalten sein müssen.

3.2.2 Rolle Distributions-Verteiler

Die Verteilung der Distribution umfasst die Bereitstellung von Programmen sowie die Integration dieser in Nutzersysteme. Jedermann darf auf beliebige

Art und Weise die Distributions-Komponenten bereitstellen. Für die automatisierte Weitergabe von Komponenten in Endsysteme ist allerdings eine vorgegebene Struktur, sogenannte Debian-Repositories, einzuhalten. Die Integration einer Komponente in ein Endsystem muss die vom Distributions-Ersteller definierten Abhängigkeiten beachten.

3.3. Tätigkeiten des Distributions-Erstellers

Ein Grossteil der Tätigkeiten des Distributions-Erstellers wird bei Debian von sogenannten Package-Maintainern durchgeführt. Maintainer kann prinzipiell jeder werden, der sich bereit erklärt ein bestimmtes Programm zu betreuen.

Ein Maintainer nimmt also ein Programm seiner Wahl, welches er der Distribution hinzufügen möchte. Dazu überprüft er als erstes die Lizenz sowie die Verfügbarkeit des Source-Codes des Programmes, und vergleicht diese mit den Debian Vorgaben entsprechend Abschnitt 3.2.1. Entsprechend der Lizenz ordnet er das Programm einer der Sektionen main, contrib oder non-free zu.

Sollten Anpassungen an dem Programm zur Integration in ein Debian-System nötig sein, so erstellt der Maintainer einen entsprechenden Patch. Alle bereits in Debian enthaltene Programme die zur Ausführung benötigt werden, müssen durch entsprechende Abhängigkeiten angegeben werden. Ausserdem erstellt er alle zur Installation benötigte Skripte. Zum Schluss werden all diese Angaben, Patche, Skripte zusammen mit dem Programm selber in einem klar definierten Dateiformat (Debian-Paket, siehe Abschnitt 4) zusammengefasst.

Um das neue Debian-Paket an den Distributions-Verteiler weiterzugeben muss dieses nur noch in das unstable Repository auf ftp-master.debian.org hochgeladen werden. Diese Aktion darf jedoch nicht jeder selber durchführen, sondern nur offizielle Debian Developer dürfen dies. Ist man selber kein Debian Developer, ist die einzige Möglichkeit einen Developer zu finden der das eigene Paket unterstützt.

Sobald das Paket in unstable verfügbar ist, kann jeder Endbenutzer, der sich für die unstable Distribution entschieden hat, dieses Paket auf seinem Rechner installieren und ausprobieren. Neben unstable gibt es auch andere Repositories: stable, testing und experimental. Debian stellt ein Bug Tracking System zur Verfügung [5]. Jeder Benutzer eines Debian-Systems darf Fehler in Programmen dort entsprechend melden. Dieses Bug Tracking System wird insbesondere dazu benutzt, um zu entscheiden ob ein Paket automatisch

von unstable nach testing verschoben wird. Eines der Kriterien ist zum Beispiel das die Anzahl der release-critical bugs kleiner oder gleich der bereits vorhandenen Version desselben Pakets in testing ist. Sind alle Kriterien erfüllt, wird das Paket von unstable nach testing verschoben. Dies wird durch das Skript britney durchgeführt, welches die *Packages* Paketliste für testing erzeugt.

Zur Koordination all dieser Vorgänge gibt es viele weitere Hilfsmittel. Zum Beispiel gibt es eine Webseite 'Work-Needing and Prospective Packages' [18], auf der Pakete gelistet werden die bereits in der Erstellung durch einen Maintainer sind, sowie verschiedene Programme die noch einen Maintainer benötigen. Eine genaue Auflistung aller benötigter Informationen für Debian Maintainer und Developer gibt es in der 'Debian Developer's Reference' [6].

3.4. Tätigkeiten des Distributions-Verteilers

Die Aktionen des Verteilers lassen sich in zwei grobe Teile unterscheiden. Zum einen werden öffentliche Repositories bereitgestellt, zum anderen werden Programme und Tools bereitgestellt, um einzelne Pakete automatisch von diesen Repositories zu beziehen, zu installieren, zu konfigurieren, zu entfernen und vieles mehr.

Öffentliche Repositories können von jedem bereitgestellt werden. Die einfachste Variante ist einen FTP-Server aufzusetzen und alle Dateien eines bereits existierenden Debian-Servers zu kopieren und selber bereitzustellen. Genaue Informationen wie man am besten einen Debian-Mirror aufsetzt findet man in der Debian Repository-HOWTO [11]. Um offizieller Debian Mirror zu werden sind allerdings bestimmte Anforderungen zu erfüllen.

Offizielle Debian Mirror werden entweder durch regelmäßiges Spiegeln aktualisiert, oder durch sogenanntes push-mirroring [10] automatisch geupdated, sobald ein neues Paket im Debian Master Server vorhanden ist. Zum Auffinden geeigneter Mirrors gibt es eine Liste der verfügbaren Debian Mirror [12].

Neben Repository-Servern im Internet werden aber auch CDs und DVDs sowohl verkauft [17] als auch zum download angeboten [8].

Ein Benutzer kann also sowohl direkt über das Internet, als auch von CDs oder DVDs auf die Debian Pakete zugreifen, und diese mittels entsprechender Programme auf seinem Rechner installieren und administrieren (siehe Abschnitt 5.1).

4. Information Viewpoint

Bei der Betrachtung der Informationsobjekte des Debian Distributions-Managements kann man generell zwischen Objekten unterscheiden die zur Automatisierung und Verwaltung des Distributions-Managements nötig sind, zwischen Objekten die über rechtliche Gegebenheiten bestimmen und den eigentlichen Objekten die für ein fertig installiertes Debian Endsystem von Interesse sind.

Lizenzen regeln die rechtlichen Gegebenheiten bezüglich der Weiterverteilung und Nutzung von Programmen. Diese Informationsobjekte sind textuelle Dokumente und natürlich nicht automatisiert auswertbar sondern von einem Menschen (meist von dem Maintainer) zu bewerten.

Programme die mittels des Distributions-Managements verteilt und verwaltet werden sollen, gibt es sowohl in Source-Code als auch in Binärer Form.

Bug Reports sind Objekte, die zur Qualitätssicherung von Programmen von Menschen eingegeben werden und durch angebbare Klassifizierungen (z.B. release-critical bug) den weiteren Lebenslauf von Paketen auf den offiziellen Debian Repositories automatisiert beeinflussen können.

Paketlisten sind Textdateien in denen Meta-Informationen zu Debian Paketen stehen. Es gibt für jede Distribution (stable, unstable, testing, experimental) jeweils drei verschiedene Paketlisten: main, contrib und non-free. Die Einsortierung hängt von der Art der Lizenz ab unter der das jeweilige Programm steht. Diese Listen enthalten jeweils die Information aus den control-Dateien der Debian Pakete (siehe 4) und ermöglichen so unter anderem eine automatisierte Abhängigkeitsauflösung.

Debian Pakete sind die grundlegenden Informationsobjekte in denen Programme zu einem Endbenutzersystem gelangen. Für jedes in einer Debian Distribution enthaltene Programm gibt es mindestens ein Debian Paket. Diese enthalten Patches, Install-Skripte, Abhängigkeiten, weitere Meta-Informationen und das Programm selber. Jedes Paket ist mit GNU ar [13] verpackt und enthält die folgenden drei Dateien:

1. **debian-binary**: Ist eine Textdatei, die nur die Versionsnummer des verwendeten Debian-Paketformats (z.B.: 2.0) enthält.
2. **data.tar.gz**: die eigentlichen Dateien, die vom

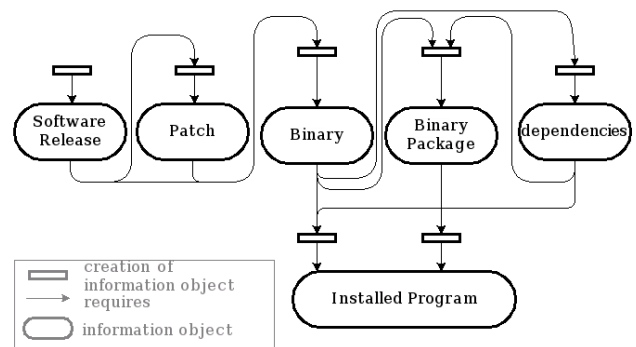
Programm benötigt werden. Dieses Tar-Archiv [15] ist genau so in einer Verzeichnisstruktur organisiert, wie die Dateien in Endsystemen abgelegt werden sollen.

3. control.tar.gz: Metadaten des Pakets

Von besonderem Interesse ist hier das **control.tar.gz** file. Es enthält eine Datei namens **control**, in der alle Meta-Informationen des Pakets gespeichert sind: Package, Source, Version, Section, Priority, Architecture, Essential, Installed-Size, Maintainer, Description. Außerdem enthält es die Abhängigkeiten, unterteilt in Depends, Recommends, Suggests, Enhances und Pre-Depends.

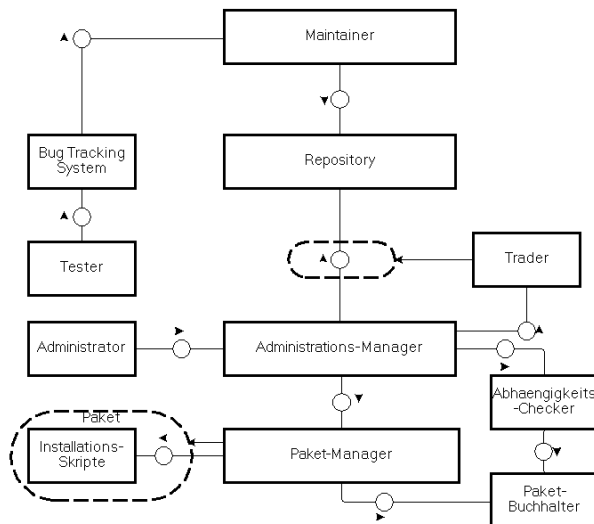
Zusätzlich enthält das **control.tar.gz** die benötigten Installations Skripte (z.B.: **preinst**, **prerm** und/oder **postrm**) und eine Checksummen-Datei (**md5sums**) mit MD5 Checksummen der in **data.tar.gz** enthaltenen Dateien.

Folgende Abbildung veranschaulicht den Prozess von der Paketbildung bis zur Installation eines Programms.



5. Computational Viewpoint

Folgende Abbildung stellt Verbindungen zwischen unterschiedlichen Systemteilen dar. Jeder Kasten ist eine funktionale Komponente. Die Striche mit einem Kreis in der Mitte stellen Kommunikationswege dar, wobei der kleine Pfeil über den Kreisen jeweils die Richtung der Client-Server Beziehung aufzeigt.



Administrations-Manager Dies ist das zentrale Tool für den Administrator um Programme über das Distributions-Management auf einem Computer zu installieren und wieder zu entfernen.

Paket-Manager Diese Komponente kann einzelne Debian Pakete in das System integrieren. Er kennt also den internen Aufbau der Pakete, und führt die ggf. enthaltenen Skripte aus. Er vergleicht die in einem Paket angegebenen Abhängigkeiten mit den bereits installierten Paketen, und kann anhand dessen z.B. entscheiden ob das gewünschte Paket überhaupt installierbar ist. Über den Paket-Manager kann zwecks Konfiguration oder Administration einzelner Programme auf die Installations-/Konfigurations-Skripte der Pakete zugegriffen werden.

Paket-Buchhalter Führt die Liste aller installierten Pakete, sowie in welchem Zustand sich das jeweilige Paket gerade befinden (z.B. "fertig installiert", oder "noch nicht konfiguriert").

Abhängigkeits-Checker Führt die Liste der automatisch beschaffbaren Pakete. Er kennt ebenfalls alle Abhängigkeiten dieser und kann somit Auskunft darüber geben ob ein Paket installiert werden kann bzw. welche weiteren Pakete dazu notwendig wären bzw. welche anderen Pakete dafür entfernt werden müssten.

Trader Kennt verschiedene Quellen von Debian Repositories und kann somit die Verbindung zu z.B. einem Debian Mirror erstellen wenn ein Debian-Paket benötigt wird.

Installations-Skripte Diese Skripte sind in Paketen enthalten (angedeutet durch die gestrichel-

te Ellipse), und können zusätzliche Installations-, Konfigurations-, Update-, Downgrade- oder Deinstallations-Routinen für das in dem Paket enthaltene Programm ausführen.

Administrator Legt fest welche Administrative Aufgabe der Administrations-Manger als nächstes durchführen soll. Er muss ebenfalls auf eventuelle Fehlermeldungen oder Konfigurations-Fragen reagieren.

Tester Ein Tester benutzt ein Debian System und meldet gefundene Fehler an das Debian Bug Tracking System.

Bug Tracking System Verwaltet Bug-Listen für alle Debian Pakete. Es stellt Interfaces für Tester und Maintainer zur Verfügung, kann Maintainer über neue Bugs informieren.

Repository Stellt Pakete und Paketlisten bereit.

Maintainer Erstellt Pakete und fügt diese dem Repository hinzu.

5.1. Verhaltens des Administrations-Managers

Bei einem typischen Installations-Auftrag führt der Installations-Manager folgende Aktionen durch:

1. Installationsauftrag entgegennehmen
2. Frage Abhängigkeits-Checker welche Pakete benötigt werden
3. Frage Trader ob es diese Pakete gibt und wo sie zu finden sind
4. Hole alle benötigten Pakete über Interface zu Repository
5. Übergebe Pakete geordnet an Paket-Manager zur Installation

Will der Administrator eine Software wieder Deinstallieren, so führt der Administrations-Manager folgende Aktionen durch:

1. Deinstallationsauftrag entgegennehmen
2. Frage Abhängigkeits-Checker welche Pakete von Paket xx abhängig sind
3. Frage ggf. Administrator ob das Paket und alle davon abhängigen Pakete tatsächlich deinstalliert werden sollen
4. Übergebe Deinstallations-Anforderungen geordnet an Paket-Manager

5.2. Interface des Administrations-Managers apt

```
Usage: apt-get [options] command
apt-get [options] install|remove pkg1 [pkg2 ...]
apt-get [options] source pkg1 [pkg2 ...]
```

```
Commands:
update - Retrieve new lists of packages
upgrade - Perform an upgrade
install - Install new packages (pkg is libc6 not libc6.deb)
remove - Remove packages
source - Download source archives
build-dep - Configure build-dependencies for source packages
dist-upgrade - Distribution upgrade, see apt-get(8)
dselect-upgrade - Follow dselect selections
clean - Erase downloaded archive files
autoclean - Erase old downloaded archive files
check - Verify that there are no broken dependencies
```

Es ist zu bemerken, daß apt [1] nicht nur den Administrations-Manager implementiert, sondern auch den Trader und den Abhängigkeits-Checker in sich vereint. Dazu speichert apt die Paketlisten der Repositories im Verzeichnis `/var/lib/apt/lists`. Die Liste der verfügbaren Repositories wird vom Administrator in der Datei `/etc/apt/sources.list` angegeben. Apt beinhaltet auch weitere Tools für unterschiedliche Aufgaben, wie z.B. `apt-cache` zum durchsuchen der verfügbaren Pakete oder `apt-cdrom` zum hinzufügen einer CD als weiteres verfügbares Repository.

5.3. Interface des Paket-Managers dpkg

Der Paket-Manager und der Paket Buchhalter wird in Debian Systemen durch das Programm `dpkg` implementiert. Für die Funktionalität des Buchalters führt `dpkg` die Liste `/var/lib/dpkg/status`, in der jedes Installierte Paket und sein Installations-Zustand vermerkt sind.

```
dpkg [<option> ...] <command>

Commands:
-i|--install <.deb file name> ... | -R|--recursive <directory>
-r|--remove <package> ... | -a|--pending
-P|--purge <package> ... | -a|--pending
-s|--status <package> ... Display package status details.
-p|--print-avail <package> ... Display available version details.
-L|--listfiles <package> ... List files 'owned' by package(s).
-l|--list [<pattern> ...] List packages concisely.
-S|--search <pattern> ... Find package(s) owning file(s).
[...]
```

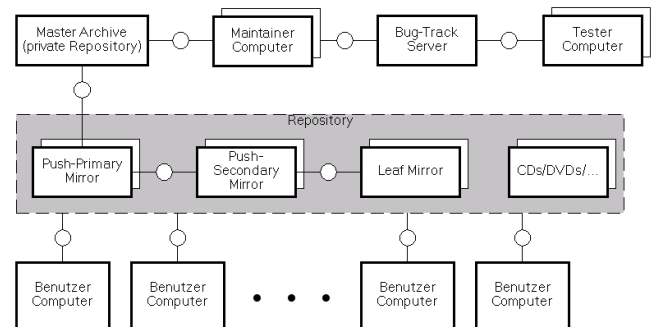
6. Engineering Viewpoint

Folgende Abbildung veranschaulicht besonders die Verteilung der Repositories. Die geschachtelten Kästchen sollen eine Vielzahl gleichartiger Nodes darstellen.

In dem in der Mitte grau hinterlegten Kasten ist die Verteilung der öffentlichen Repositories dargestellt.

Links oben befindet sich der Master Server, der mittels Push-Mirroring die Push-Primary Server mit neuen Paketen versorgt. Die Repositories bilden somit eine Hierarchie, angefangen mit dem Master Server werden die Repositories zu Primary Servern und von dort zu Secondary Servern verteilt. Die Leaf Server updaten sich selber, greifen dabei aber sowohl auf Primary als auch auf andere Secondary oder Leaf Server zurück.

Jeder Benutzer kann praktisch auf beliebige Repositories zugreifen. Meistens wird ein geografisch in der Nähe gelegener Mirror als Repository-Server gewählt.



7. Konklusion

Das Distributions-Management von Debian ist somit ein, aus Sicht des Endbenutzers, voll automatisiertes System zur Beschaffung und Installation vieler verschiedener Programme unterschiedlichster Herkunft.

Ein sich daraus ergebender Vorteil ist eine stabile Koexistenz vieler verschiedener Programme auf einem Endsystem, was, im Gegensatz dazu, bei Systemen ohne ein vergleichbares Distributions-Management, weder zentral koordiniert noch getestet wird.

Als weiteren Vorteil ergibt sich die automatisierte Aktualisierung aller über dieses System installierter Programme durch einen einzigen Befehl. Besonders gut ist dies für sicherheitskritische Systeme, die mit automatisierten Security-Updates immer auf den neuesten Sicherheits-Standard gebracht werden können. Im Gegensatz zu Systemen ohne Distributions-Management bleiben so alle Anwendungen (insbesondere auch alle installierten Server-Dienste) automatisch up-to-date, ohne jedes Programm händisch einzeln updaten zu müssen.

Speziell dies hier beschriebene Distributions-Management System von Debian ist jedoch kein von Anfang an *durchgeplantes* Software-System, sondern ein über die Jahre an verschiedenen Enden gleichzeitig gewachsenes System. Möglich wurde das alles erst mit der Verbreitung vieler und schneller Internetzugänge, da plötzlich sehr große Datenmengen an sehr viele Benutzer über das Internet automatisiert verteilt

werden.

Unsere persönliche Erfahrung mit Debian ist "Wer sich einmal daran gewöhnt hat, will es nie wieder missen!"

Literatur

- [1] *APT HOWTO*. <http://www.debian.org/doc/manuals/apt-howto/>.
- [2] *The Artistic License*. <http://www.opensource.org/licenses/artistic-license.php>.
- [3] *The BSD License*. <http://www.opensource.org/licenses/bsd-license.php>.
- [4] *Debian*. <http://www.debian.org>.
- [5] *Debian bug tracking system*. <http://www.debian.org/Bugs/>.
- [6] *Debian Developer's Reference*. <http://www.debian.de/doc/manuals/developers-reference/index.en.html>.
- [7] *The Debian Free Software Guidelines (DFSG)*. http://www.debian.org/social_contract#guidelines.
- [8] *Debian GNU/Linux on CDs*. <http://www.debian.org/CD/>.
- [9] *Debian Policy Manual*. <http://www.debian.org/doc/debian-policy/>.
- [10] *Debian Push Mirroring*. http://www.debian.org/mirror/push_mirroring.
- [11] *Debian Repository - HOWTO*. <http://www.debian.org/doc/manuals/repository-howto/repository-howto>.
- [12] *Debian worldwide mirror sites*. <http://www.debian.org/mirror/list>.
- [13] *GNU Development Tools ar(1)*. <http://www.iti.cs.tu-bs.de/cgi-bin/UNIXhelp/man-cgi?ar+1>.
- [14] *GNU General Public License*. <http://www.gnu.org/copyleft/gpl.html>.
- [15] *GNU tar manual*. <http://www.gnu.org/software/tar/manual/index.html>.
- [16] *Open Distributed Processing - Reference Model*. ISO/IEC 10746.
- [17] *Vendors of Debian CDs*. <http://www.debian.org/CD/vendors/>.
- [18] *Work-Needing and Prospective Packages*. <http://www.debian.org/devel/wnpp/>.