D09-QM-2 Release Management von Software-Modulen und Dokumenten der Open-Source-„PersoApp“

Editor:	Jens Kubieziel (AGETO)
Prüfung:	Sven Wohlgemuth (TU Darmstadt/CASED)
Typ:	[TECHNISCHER BERICHT]
Projekt:	„PersoApp“
Version:	1.0
Datum:	25. Juni 2013
Status:	[FREIGABE]
Klasse:	[ÖFFENTLICHKEIT]
Datei:	D09-QM-2 Release Management von Software-Modulen und Dokumenten der Open-Source-PersoApp.doc

Zusammenfassung

Dieses Dokument liefert eine Prozessbeschreibung für das Releasekonzept. Es wird auf Funktionstests, die Freigabe von Releases und die Releases für verschiedene Betriebssysteme eingegangen.

Konsortialleitung:
Prof. Dr. Ahmad-Reza Sadeghi und Dr. Sven Wohlgemuth
System Security Lab, TU Darmstadt/CASED, Mornewegstr. 32, 64293 Darmstadt
Tel.: +49-6151-16-75561 Fax: +49-6151-16-72135
E-Mail: persoapp@trust.cased.de Web: https://www.persoapp.de

Nutzungslizenz

Mitglieder des Konsortiums

1. **AGETO Service GmbH**, Deutschland
2. **Center for Advanced Security Research Darmstadt (CASED)**, Deutschland
3. **Fraunhofer Institut für Sichere Informationstechnologie (SIT)**, Deutschland
4. **Technische Universität (TU) Darmstadt**, Deutschland

Versionen

Version	*Datum*	*Beschreibung (Editor)*
0.1	2013-06-21	Initiale Version des Dokuments (Jens Kubieziel)
0.2	2013-06-22	Beschreibung zu Microsoft Windows und Mac OS X (Jens Kubieziel)
0.3	2013-06-25	Sicherheitstests, Release Management für GNU/Linux (Jens Kubieziel)
0.4	2013-06-25	Fertigstellung zum Review (Jens Kubieziel)
1.0	2013-06-25	Überarbeitung zur Freigabe (Jens Kubieziel)

Autoren

Autoren	*Beiträge*
Jens Kubieziel (AGETO)	Initiale Version
Philipp Holzinger (Fraunhofer SIT)	Sicherheitstests, Release Management für GNU/Linux

[1] http://creativecommons.org/licenses/by-sa/3.0/

Inhaltsverzeichnis

1 Ziel und Zweck des Dokumentes

Das Dokument beschreibt den Prozess der Freigabe neuer Releases. Es wird auf funktionale wie auch Sicherheitstests eingegangen. Weiterhin erfolgt eine Festlegung wie die Software für verschiedene Betriebssysteme bereitgestellt wird. Zu den Systemen gehört Microsoft Windows, verschiedene Distribution von GNU/Linux sowie Mac OS X.

2 Anwendungsbereich

Die beschriebenen Prozesse finden bei den Major Releases Anwendung. Sie geben eine Richtschnur vor, welche Tests durchzuführen sind und auf welche Weise das ausführbare Programm für den Endanwender zur Verfügung gestellt wird.

3 Abkürzungen und Begriffsdefinition

3.1 Abkürzungen

Abkürzung	*Erläuterung*
CASED	Center for Advanced Security Research Darmstadt
CC	Creative Commons
eID	Elektronische Identität
GCJ	GNU Compiler for Java
GNU	GNU’s Not UNIX
JAR	Java Archive
JDK	Java Development Kit
JNLP	Java Network Launching Protocol
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MIME	Multipurpose Internet Mail Extensions
MOTU	Master of the Universe
nPA	Neuer Personalausweis
PE	Portable Executable
SIT	Sichere Informationstechnologie (s. Fraunhofer SIT)
TR	Technische Richtlinie
TU	Technische Universität

URL	Uniform Resource Locator
XML	eXtensible Markup Language

3.2 Begriffsdefinitionen

Begriff	*Definition*
GNU	GNU ist ein rekursives Akronym (siehe Abkürzungen) und bezeichnet ein Projekt, welches ein Freies Betriebssystem entwickeln will.
GNU/Linux	GNU/Linux ist die offizielle Bezeichnung für den Kern des GNU-Betriebssystems. Dies wird oft mit Linux gekürzt verwendet.
MIME	MIME dient dazu, den Typ der zu übermittelnden Daten festzulegen. Dies wird verwendet, um Bilder, Audioinhalte und anderes über textbasierte Medien (E-Mail, Web etc.) zu übertragen.
MOTU	Die Software-Archive der Distribution Ubuntu heißen u.a. universe und multiverse. Spezielle Mitglieder aus der Ubuntu-Community pflegen die Archive. Diese tragen die Abkürzung MOTU.

4 Zuständigkeiten und Verantwortlichkeiten

Dieses Dokument wurde in gemeinschaftlicher Zusammenarbeit vom Fraunhofer-Institut für Sichere Informationstechnologie (SIT) und der AGETO Service GmbH erstellt. Die weiteren Konsortialpartner wurden in Form eines Reviews mit der Möglichkeit für Kommentare und Änderungswünsche indirekt daran beteiligt.

5 Release-Management von Software-Modulen und Dokumente der Open-Source-„PersoApp“

5.1 Funktionale Tests

Funktionale Tests bzw. Funktionstests haben ihren Ursprung in der Fertigung industrieller Güter. Unter dem Stichwort „Verifikation“ wird dort versucht, die Frage zu beantworten, ob das zugrundeliegende System entsprechend der Planung korrekt angefertigt wurde. Für Software ist es ebenso wichtig zu erfahren, ob die entstandene Software der Spezifikation entspricht.

Funktionale Tests sind dabei nur ein Teil des Testprozesses. Im Rahmen einer Softwareentwicklung, die dem testgetriebenen Paradigma folgt, beginnen Tests schon in frühen Stadien, wie der Spezifikations- oder Entwurfsphase. Dadurch können früh Fehler bzw. falsche Annahmen identifiziert und korrigiert werden. Die Früherkennung hilft, die Kosten für die Softwareentwicklung zu minimieren. Üblicherweise steigt der Aufwand für die Korrektur von Fehlern, je später im Entwicklungsprozess diese ge-

funden werden. Der Aufwand ist am höchsten, wenn die Software schon im Produktiveinsatz beim Kunden ist.

Es ist daher zu empfehlen, dass spätestens bei der Programmierung der Software Test-Techniken zum Einsatz kommen. Das heißt, mittels Unit-Tests werden kleine Programmteile getestet. Diese Tests werden meist automatisch durchgeführt und sichern Korrektheit der getesteten Module. Im Anschluss können Integrationstests stehen. Dabei wird das Zusammenspiel von Modulen, die in Verbindung stehen, getestet. Eine Methode für derartige Tests sind die Funktionstests. An dieser Stelle der Integrationstests ist es die Aufgabe von funktionalen Tests, festzustellen, ob die Software die richtige Komponente benutzt. Das Gesamtbild verfügbarer Tests wird durch Systemtests und Akzeptanztests abgerundet. Die Aufgabe von Systemtests ist es, dass System als Einheit zu testen. Das heißt, die Software wird auf einem Rechner mit den im Endeinsatz üblichen Einheiten (Hard-, Software etc.) getestet. Der Akzeptanztest wird schließlich vor Ort beim Kunden durchgeführt und stellt meist den Abschluss der Produktentwicklung dar.

Der Ausgangspunkt für funktionale Tests ist die Spezifikation der Software. Hingegen finden der Quellcode und die Struktur des Programms keine Berücksichtigung. Anhand der Spezifikation wird in einem ersten Schritt versucht, gültige Ein- und Ausgabedaten zu ermitteln. Unter Umständen sind in beiden eine Vielzahl gleichartigen Ein- und Ausgaben zu finden. Daher erfolgt im nächsten Schritt eine Bildung von Klassen. Ein- und Ausgaben, die nach ihrer Form bzw. Ergebnis äquivalent sind, werden als gleich betrachtet. Im späteren Testverlauf kann eine zufällige Auswahl eines Wertes aus einer Klasse von Daten erfolgen. Die gebildeten Klassen umfassen sowohl gültige wie auch ungültige Werte. Die ungültigen Werte sollen eine Gegenprobe darstellen und ermitteln, ob Eingaben, die nicht funktionieren sollen, tatsächlich nicht funktionieren.

Beim eigentlichen Funktionstest wird ein Wert aus einer Äquivalenzklasse gewählt und als Eingabe verwendet. Es empfiehlt sich, Werte „vom Rand“ der Klasse zu wählen. Falls beispielsweise die Postleitzahl ein Eingabewert ist, so liegt der Eingabebereich zunächst zwischen 00000 und 99999.[2] Zahlen, die mehr oder weniger als 5 Stellen lang sind, sowie Eingaben, die nichtnumerische Werte enthalten, sind ungültig. Ein einfacher funktionaler Test könnte dann Eingaben testen, die 4 Stellen lang sind und nicht mit einer Null beginnen, mehr als 5 Stellen lang sind und aus führenden Nullen bestehen oder Werte herausgreifen, die nahe bei 00000 bzw. 99999 liegen. Im Rahmen eines tiefergehenden Funktionstest kann die Frage stehen, ob eine Postleitzahl korrekt einem entsprechenden Ort zugewiesen wird. Hierbei sollten insbesondere Eingabewerte gewählt werden, bei denen es in der jüngeren Vergangenheit Änderungen gab.

[2] Dabei sei vorausgesetzt, dass nur Postleitzahlen nach dem aktuellen Vergabesystem und nur deutsche Postleitzahlen betrachtet werden.

5.2 Freigabe von Releases

Im Rahmen des Projektes „PersoApp“ sind ein Pre-Release und fünf Major Releases geplant. Das Pre-Release stellt den proprietären eID-Client der AGETO Service GmbH bereit. Dieser dient der Öffentlichkeit zu Testzwecken. Der Client wurde sowohl von der AGETO Service GmbH wie auch vom Fraunhofer Institut für Sichere Informationstechnologie in Augenschein genommen. Auf der Grundlage von Systemtests wurde sichergestellt, dass der Client unter den Betriebssystemen Microsoft Windows, GNU/Linux und Mac US X fehlerfrei gestartet und ausgeführt werden kann. Der eID-Client steht auf der Webseite http://www.persoapp.de/ als JAR-Datei zum Download zur Verfügung.

Weitere Releases im Rahmen des Projektes „PersoApp“ verfolgen einen definierten Prozess. Im Dokument „D08-QM-2 Operative Planung und Durchführung von Reviews und Release-Updates“ sind die beiden Phasen, Entwicklungs- und Testphase, beschrieben. Demnach werden im Rahmen der Entwicklungsphase neue Features eingebaut, Fehler behoben etc. Im Anschluss daran steht die Testphase. In dieser Phase finden funktionale Tests und Reviews (Inspektion von Major-Release-Updates) statt. Insbesondere in der Inspektionsphase könnten Fehler gefunden werden. Es liegt in der Aufgabe des Leiters der Inspektion, die Ergebnisse des Reviews zu kommunizieren. Wenn die Inspektion mit einem positiven Ergebnis zu Ende gegangen ist, kann das Release erfolgen.

Im Versionskontrollsystem wird die korrekte Version des Quellcodes markiert (getaggt). Ausgehend von dieser Markierung erfolgt ein Export des Quellcodes in ein Archiv. Dazu ist das Format „tar.gz“ zu empfehlen. Dieser Quellcode dient als Grundlage für die Anfertigung des Binärcodes für die Betriebssystem Microsoft Windows, GNU/Linux und Mac OS X.

Nach der Fertigstellung werden die erzeugten Dateien auf die Webseite des Projektes – http://www.persoapp.de/ – hochgeladen. Weiterhin empfiehlt es sich eine kurze Zusammenfassung der Änderungen seit dem letzten Major Release zu erstellen und ebenfalls in die Webseite aufzunehmen. Für den Fall, dass spezielle Repositorys zur Verfügung gestellt werden, so sind diese ebenso mit der aktuelle Software zu ergänzen.

Abschließend sollte eine Information der Öffentlichkeit über die im Dokument „D05-QM Dokumentationsanforderungen“ definierten Kanäle erfolgen.

5.3 Sicherheitstests

Im Rahmen des Release-Prozesses von „PersoApp“ werden neben den funktionalen Tests auch Sicherheitstests durchgeführt. Hierbei handelt es sich vorwiegend um automatische Tools, die auf den Quelltext angewendet werden können. Während der Entwicklungsphase können die Entwickler aus dem Kernteam und der Community Patches mit Fehlerkorrekturen und neuen Features einreichen. Nach einem Feature-Freeze und funktionierenden funktionalen Tests, wird der Quellcode mit Hilfe von automatischen Tools, wie z.B. Findbugs, Checkstyle und SonarQube gescannt. **Abbildung 1** beschreibt das zeitliche Vorgehen.

Nach der Einberufung des Feature-Freeze durch den Release Manager, initiiert der der Qualitätsmanager den Beginn der Sicherheitstests. Diese können weitestgehend auch parallel zu den funktionalen Tests durchgeführt werden. Der Qualitätsmanager delegiert hierzu einzelne Testaufgaben an Entwickler im Kernteam, an Modulverantwortliche oder die Community. Die Ergebnisse der automatischen Tools werden den für die Änderungen verantwortlichen Entwicklern mitgeteilt.

Gefundene Abweichungen von den Qualitätskriterien aus dem Dokument „D02-QM Qualitätskriterien: Aufbau, Messgrößen und Bewertung“ sollten von den Entwicklern schnell beseitigt werden. Diese Abweichungen sind jedoch nicht als sehr kritisch einzustufen und somit kann der Qualitätsmanager im Einzelfall auch entscheiden, dass die Abweichungen ohne Korrektur akzeptiert werden.

Sollte die Anwendung von statischen Codescannern jedoch dazu führen, dass kritische Sicherheitslücken gefunden werden, sind diese unverzüglich von den verantwortlichen Entwicklern zu schließen. Die Nichtbehebung solcher Lücken führt zum Entfernen des entsprechenden Quelltextes oder notfalls zum Stopp des Releases bis ein anderer Entwickler, z.B. aus dem Kernteam, die Sicherheitslücke geschlossen hat.

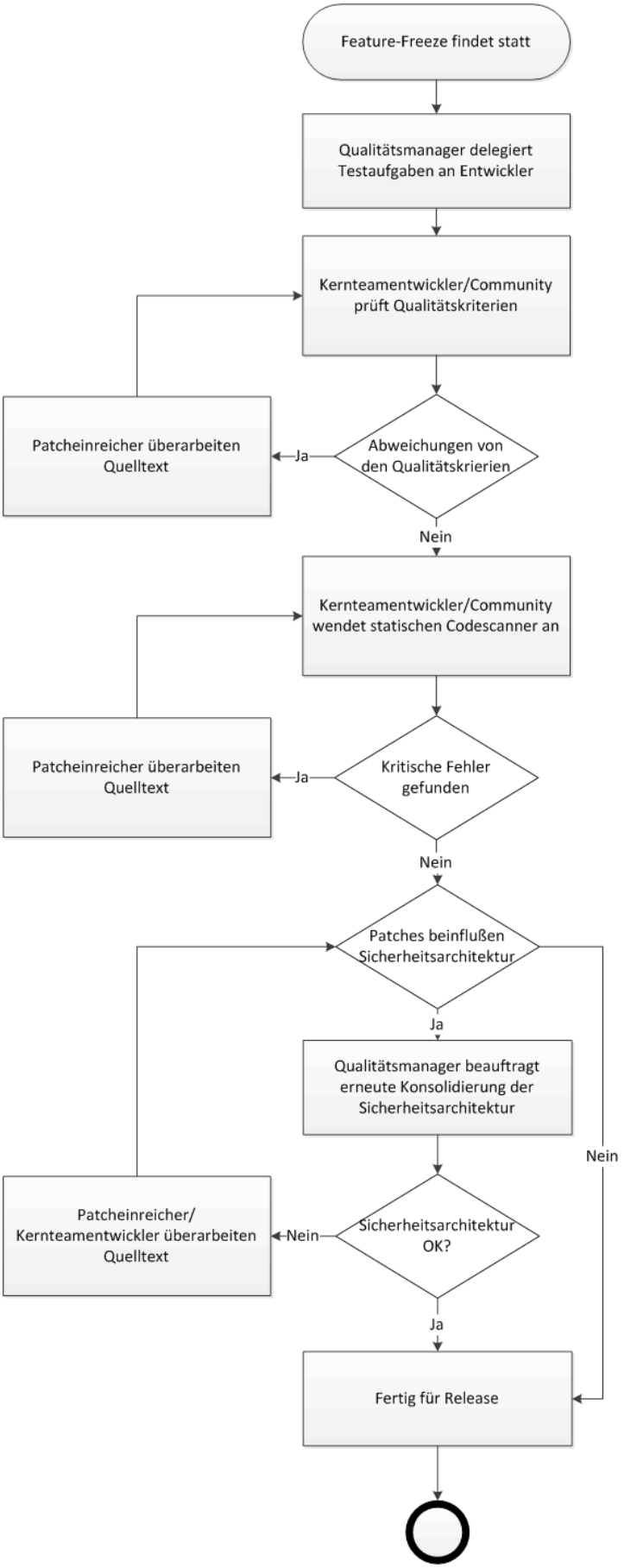

Abbildung 1: Prozess für Sicherheitstests

Abhängig davon, wie viele Patches eingereicht werden und wie umfangreich diese sind, kann der Qualitätsmanager eine erneute Konsolidierung der Sicherheitsarchitektur beim Kernentwicklerteam beauftragen. Hierzu evaluiert das Kernentwicklerteam die in „D06-QM Architekturkonzept der Open-Source-Core“ erstellte Sicherheitsarchitektur mit den durch die Patches eingereichten Änderungen. Besonders interessant bei dieser erneuten Evaluierung sind die durch die Änderungen betroffenen Sicherheitsmechanismen oder Assets. Sollte die Wirksamkeit der Sicherheitsmechanismen durch die Patches nicht beeinträchtigt worden sein und sind auch keine neuen Mechanismen erforderlich, so ist alles in Ordnung. Wenn jedoch die Wirksamkeit der Sicherheitsmechanismen beeinträchtigt wird, so wägt das Kernentwicklerteam ab, unter welchen Umständen die Patches akzeptiert werden können und gibt daraus resultierende Änderungswünsche an die verantwortlichen Entwickler weiter. Erst wenn diese Änderungswünsche umgesetzt worden sind und der betroffene Teil der Sicherheitsarchitektur danach erneut konsolidiert wurde, werden die Patches für ein Release akzeptiert.

5.4 Erstellung von Linux-Releases

Im Rahmen der Releases zur „PersoApp“ werden wir Pakete für verschiedene Linux-Distributionen erstellen. Nutzern einer „PersoApp“-Software soll ihre Installation so einfach wie möglich gemacht werden. Für die folgenden Distributionen wird es fertig installierbare Pakete geben: Ubuntu/Debian, Fedora, Gentoo. Diese Distributionen wurden auf Basis von „Distrowatch.com“ als Major Linux Distribution identifiziert, die auch zugleich die wichtigsten Paketmanagementsystem („APT/DEB“, „YUM/RPM“ und „Ebuild“) verwenden. Ergänzend werden alle Pakete jeweils für 32- und 64-Bit-Systeme bereitgestellt. Für jede Distribution wird ein verantwortlicher Distributionsmanager von der Projektleitung benannt.

Für den in Abständen von 6 Monaten geplanten Quellcode Major Release sollen innerhalb von kürzester Zeit entsprechende Pakete für o.g. Linux-Distributionen veröffentlicht werden. Vor Bereitstellung der einzelnen Pakete muss vom zuständigen Distributionsmanager sichergestellt werden, dass sich die Software in der jeweiligen Umgebung problemlos installieren lässt. Dazu werden jeweils innerhalb einer Woche nach der Quellcode-Freigabe entsprechende Release-Candidates (RC) für o.g. Distributionen erstellt und der Community zum Download in Form von Unstable- oder Testing-Paketen bereitgestellt. Die Community hat dann zwei Wochen Zeit, über das Bug-Tracking-System Feedback zu geben. Werden bis zum Ablauf dieser Testphase keine Bugs gemeldet, werden die entsprechenden Pakete als stable markiert und in Repositorys entsprechend bereitgestellt. Andernfalls wiederholt sich o.g. Prozess.

Außerplanmäßige Releases können durch distributionsspezifische Änderungen, veränderte Abhängigkeit etc. notwendig werden. In solchen Fällen wird analog zu den Major Releases verfahren.

Um die Integrität fertiger Pakete zu garantieren, werden die Pakete signiert. Dazu wird vom Kernentwicklerteam ein GPG-Schlüsselpaar generiert. Alle Pakete werden mit demselben Schlüssel signiert, welcher unter der Obhut des Release Managers steht. Das Schlüsselpaar wird mit dem Algorithmus RSA erzeugt und hat eine Länge

von 2048 Bit bei einer Gültigkeit von 2 Jahren. Die zugehörigen öffentlichen Schlüssel werden auf der Projektseite veröffentlicht.

5.4.1 Gentoo Linux

Das Gentoo-Paketsystem „Portage“ basiert auf sogenannten „Ebuild“-Dateien, welche beschreiben wie Software von der Community (Open-Source-Software) oder Firmen auf dem System installiert wird. Im Rahmen des PersoApp-Projektes wird mindestens ein solches „Ebuild“ erstellt werden, mit dem sich die Software installieren lässt. Da Gentoo eine Quellcode-basierte Distribution ist, könnte auch die Installation aus dem Quellcode statt einer Installation mit Binärcode angeboten werden.

Langfristig wird angestrebt, das vom Projekt erstellte „Ebuild“ in das Gentoo-Projekt einfließen zu lassen, wodurch es dann Gentoo-Benutzer ohne Mehraufwand direkt installieren können. Daher wird der erste Schritt sein, das „Ebuild“ auf http://overlays.gentoo.org zu veröffentlichen. Diese vom Gentoo-Projekt verwaltete Seite bietet Entwicklern die nötige Infrastruktur zum Verteilen von „Ebuilds“ an Benutzer. Die Benutzer können eine „PersoApp“-Software von dort in ihr System installieren, indem sie das „PersoApp“-Projekt auf „overlays.gentoo.org“ in ihrem System bekannt machen. Auf dieser Webseite wird das „Ebuild“ in einem Versionskontrollsystem bereitgestellt. In ihm wird auf die Adresse zum Download der Software verwiesen. Die Software wird auf der „PersoApp“-Projektseite als ein „tar.gz“-Datenarchiv mit der Namenskonvention: „PersoApp-VERSION.tar.gz“ zur Verfügung gestellt.

Getestet wird das „Ebuild“ zum einen von dem Gentoo-Distributions-Manager selbst und durch die Community während der Betaphase. Die Betaphase erstreckt sich vom Feature-Freeze vor einem Release bis zum Taggen des Releases durch den Release-Manager. Die Tests umfassen u.a. folgende Punkte:

- Kann das Release von der Webseite heruntergeladen werden?
- Sind die Checksummen und Dateigrößen korrekt?
- Kompiliert der Quelltext korrekt? (Quelltext „Ebuild“)
- Befinden sind nach der Installation alle Dateien an den korrekten Orten im System?
- Deinstallation: Sind alle installierten Dateien gelöscht worden?

Fehler, die während des Testens gefunden werden, können direkt im offiziellen Gentoo-Bugtracker vermerkt werden. Dies dient der Nähe zum Gentoo-Projekt und vereinfacht eine eventuelle Übernahme des „PersoApp-Ebuilds“ in die offizielle Paketverwaltung.

Die Beta-Releases zum Testen werden im Testing-Zweig der Gentoo-Paketverwaltung („~arch“) veröffentlicht und werden „stable“, sobald der Release Manager das Release markiert (getaggt) hat und keine offenen Fehler im Bugtracker eingetragen sind.

Neben der „Ebuild“-Datei selber muss der Gentoo-Distributions-Manager eine so genannte Manifest-Datei erzeugen, welche die Dateigröße des „PersoApp“-Archivs

(tar.gz Datei) und dessen Hashwerte (u.a. MD5, SHA1, etc.) enthält. Diese Manifest-Datei wird mit dem GPG-Schlüssel signiert.

Der Ablauf der Paketerstellung für Gentoo ist Folgender:

- Release Manager bestimmt den Distributions-Manager für Gentoo
- Release Manager beruft Feature-Freeze ein
- Gentoo Distributions-Manager erstellt erstes Ebuild („Beta-Ebuild“)
- Release Manager taggt Release
- Gentoo Distributions-Manager passt das „Ebuild“ an letzte Änderungen und Fehlerkorrekturen an („Final-Ebuild“)
- Gentoo Distributions-Manager veröffentlicht das „Ebuild“ auf „overlays.gentoo.org“
- Release Manager teilt das Release dem Webmaster mit, der es dann auf der Webseite veröffentlicht

5.4.2 Fedora Linux

Fedora nutzt ein auf dem „RPM Package Manager“ basierendes Paketsystem, welches Programme und Bibliotheken als RPM-Pakete verwaltet. Auf das Paket-Repository kann unter Fedora über die Programme „yum“ oder „PackageKit“ zugegriffen werden. Mit beiden Programmen können RPM-Pakete gesucht, installiert, erneuert und deinstalliert werden. Das Programm „PackageKit“ besitzt dabei, im Gegensatz zu „yum“, eine grafische Oberfläche. Bei Fedora werden die RPM-Pakete in verschiedenen Repositories vorgehalten. Das offizielle Repository wird dabei standardmäßig bei der Installation von Fedora Systemen eingerichtet, weitere Repositories können vom Nutzer hinzugefügt werden.

Es werden zwei Arten von Paketen unterschieden: RPM-Pakete enthalten kompilierte Programme (Binaries); SRPM-Pakete enthalten den zugehörigen Quellcode. Das Erstellen der Pakete erfolgt in der Regel direkt aus dem Source Code. RPM-Pakete enthalten eine „.spec“-Datei, die Informationen über die gepackte Software enthält. Zum Einem sind hier Meta-Informationen über die Software selbst, wie Name, Version, Release-Nummer vorhanden, zum anderen Informationen über Abhängigkeiten und den Ablauf der Paketerstellung. Eigene RPM-Pakete können über das Programm „rpmbuild“ aus einer solchen .spec-Datei erstellt werden.

Zur Bereitstellung einer „PersoApp“-Software als binäres RPM-Paket für Fedora-Systeme bieten sich zwei Möglichkeiten an. Zu Einen die Integration in die offiziellen Paket-Quellen, zum anderen die Bereitstellung der „PersoApp“-Software in einem unabhängig von der Fedora-Distribution gehosten Repository. Langfristig wird angestrebt, das vom Projekt erstellte RPM-Paket für die „PersoApp“-Software in das offizielle Repository der Fedora-Distribution einfließen zu lassen. Zur Integration neuer Pakete in das offizielle Repository sind allerdings einige Hürden hinsichtlich Lizenzen, Paketstruktur etc. zu überwinden.

5.4.2.1 Offizielles Repository

Um ein RPM-Paket im offiziellen Fedora Repository zu veröffentlichen, muss es zunächst einen „Review Prozess“ durchlaufen. Dieser definiert zwei Rollen, den „Contributor“, der neue Software veröffentlichen möchte und den „Reviewer“, der diese Software prüft. Zusätzlich können sich auch andere Personen in den Review Prozess einschalten und auf Fehler aufmerksam machen. Der vereinfachte Ablauf des Reviewprozesses ist wie folgt:

- Der „Contributor“ lädt das RPM-Paket mit Source-Code (SRPM) auf einen beliebigen Webserver.
- Der „Contributor“ beantragt das Review im Bugtracking System von Red Hat.
- Der „Reviewer“ nimmt sich ein ausstehendes SRPM-Paket und testet dieses. Dieser Test kann zum Teil automatisiert ablaufen und prüft unter anderem die folgenden Punkte:
    - Kompiliert das Programm?
    - Werden Richtlinien hinsichtlich Name, Lizenz und Paketaufbau eingehalten?
    - Sind Abhängigkeiten korrekt und z.B. keine überflüssigen Bibliotheken vorhanden?
- Der „Reviewer“ merkt alle gefundenen Fehler an, welche der „Contributor“ im Anschluss lösen muss.

Nach erfolgreich durchlaufenem Review-Prozess kann das SRPM-Paket dann in mehreren Schritten in das nächste Fedora Release überführt werden.

5.4.2.2 Externes Projekt-Repository

Ein weiterer Weg zur kurzfristigen Bereitstellung einer „PersoApp“-Software als binäres Paket für Fedora-Systeme führt über die Erstellung eines selbst gehosteten Repositories, das von Fedora-Nutzern je nach Bedarf eingebunden werden können. Dazu sind folgende Schritte nötig:

- Anlegen eines Verzeichnises für das Repository „# mkdir /var/www/repository/public/“
- Kopieren aller gewünschten RPMs in das neue Verzeichnis „# cp persoapp.rpm /var/www/repository/public/“
- Repository-Metadaten (xml-rpm-metadata) mit dem Dienstprogram „createrepo“ aus dem Paket „createrepo“ generieren „# createrepo /var/www/repository/public/“
- Exportieren des neuen Repositories auf einem Webserver, z.B. mittels einer V-Host-Konfiguration für Apache mit Document-Root „/var/www/repository“
- Bereitstellen des öffentlichen GPG-Keys, mit dem das RPM-Paket signiert wurde

Um ein solches Repository in ein Fedora-System einzubinden, muss auf dem Zielsystem lediglich eine entsprechende Repository-Konfigurationsdatei angelegt werden, z.B. „/etc/yum.repos.d/persoapp.repo“ mit folgendem Inhalt:

[myrepo]

name=PersoApp Repository

baseurl=http://externalhost.com/repository/

gpgcheck=1

gpgkey=http://externalhost.com/publicpgp.key

enabled=1

5.4.3 Debian/Ubuntu

Debian basierte Linux Distributionen, zu denen auch Ubuntu zählt, nutzen die Paketverwaltung "Advanced Package Tool" (APT). Auf die entsprechenden Repositorys kann z.B. über die Programme „apt-get“ und „apt-cache“ auf der Kommandozeile sowie über grafische Programme wie synaptic oder den „Ubuntu Software-Center“ zugegriffen werden, um Pakete zu suchen, zu installieren oder zu entfernen. Debian-basierte Distributionen unterscheiden sich im Kern durch jeweils unterschiedliche mit einer für die jeweilige Distribution spezifischen Paketauswahl gefüllten Standard-Repositorys. Diese Repositorys bieten unter anderem den Vorteil, GPG-signierte Pakete sicher verteilen zu können.

Einzelne Debian-Pakete können entweder aus den Repositorys oder auch direkt installiert werden. Die Pakete liegen in einem für Debian spezifischen Format vor, besitzen die Dateinamenserweiterung „.deb“ und bestehen aus drei Bestandteilen:

- „debian-binary“: Versionsinformation des Pakets
- „control.tar.gz“: Metainformationen zum Paket
- „data.tar.gz“: Eigentliche Programmdateien und ein weiteres Unterverzeichnis „debian“ mit Informationen zur Paketerstellung

Die Erstellung eigener Debian-Pakete erfolgt mittels des Tools dpkg-buildpackages. Es wird empfohlen, den Buildprozess in einer speziellen pBuilder-Umgebung stattfinden zu lassen.

5.4.3.1 Offizielles Repository

Um ein Paket in den offiziellen Ubuntu-Repositorys zu veröffentlichen, gibt es zwei Möglichkeiten. Zum einen kann man das Paket zunächst in der Basis-Distribution Debian veröffentlichen und abwarten, bis es auch in der von Debian abgeleiteten Distribution „durchsickert“ wird (upstream). Alternativ ist auch eine ausschließliche Veröffentlichung in den Ubuntu-Repositorys möglich.

Bei der Ubuntu-Distribution wird bei den Repositorys zwischen Paketquellen, Archiven und Komponenten unterschieden. Die offizielle Paketquelle von Ubuntu gliedert sich in mehrere Archive:

- Ein Haupt-Archiv (z.B. „precise“), benannt nach der jeweiligen Ubuntu-Version, welches nicht mehr geändert wird
- Ein Archiv mit Sicherheitsaktualisierungen (z.B. „precise-security“)
- Ein Archiv für aktualisierte Pakete (z.B. „precise-updates“)
- Ein Archiv für Backports (z.B. „precise-backports“)
- Ein Archiv für vorgeschlagene Pakete (z.B. „precise-proposed“)

Hierbei sind die ersten drei Listen standardmäßig als aktive Quellen für den Paketmanager eingerichtet. Die anderen Listen sowie weitere Paketquellen können je nach Bedarf vom Nutzer hinzugefügt werden.

Es werden weiterhin vier Komponenten unterschieden. Einzelne Pakete sind dabei jeweils genau einer dieser Komponenten zugeordnet. Die Komponenten 'main' und 'restricted' werden von den Ubuntu-Entwicklern selbst; die Komponenten 'universe' und ’multiverse' werden von so genannten MOTUs aus der Community verwaltet. 'main' und 'universe' sind dabei Software-Komponenten mit freien Lizenzen, 'restricted' und 'multiverse' hingegen besitzen keine zu Debian kompatiblen Lizenzen.

Für letzteres muss ein Paket einen Review Prozess durchlaufen, bei dem zunächst zwei Reviewer das Paket akzeptieren müssen. Anschließend kann das Paket je nach Lizenz in die 'universe' oder 'multiverse' Komponenten aufgenommen werden.

Bei einer Veröffentlichung im Debian-Repository wird ein Paket automatisch nach einer gewissen Zeit auch in die Ubuntu-Distribution übernommen. Allerdings können Pakete nur von Debian-Entwicklern direkt dorthin hochgeladen werden. Über so genannte Sponsoren oder als Debian-Betreuer gibt es jedoch auch weitere Wege, Pakete in die Debian Paketverwaltung einzustellen.

5.4.3.2 Externes Projekt-Repository

Neben den offiziellen Quellen können Entwickler auch so ein genanntes "Personal Package Archive" (PPA) erstellen. Solche PPAs können dann bei Bedarf von Ubuntu-Nutzern zu ihren Paketquellen hinzugefügt werden, um so einfach die Software und ggf. folgende Updates beziehen zu können. PPAs können von Entwicklern nach Erstellen und Verifizieren eines Kontos auf https://launchpad.net/ erstellt werden. Hier kann dann der Quellcode für Pakete hochgeladen werden, welcher dann automatisch in Pakete für verschiedene Architekturen umgewandelt wird. Im Anschluss werden Links kreiert, welche die Anwender dann in ihre persönlichen Paketquellen eintragen können.

Längerfristig ist es wünschenswert, ein Debian-Paket für die PersoApp in den offiziellen Ubuntu-Repositorys zu haben. Kurzfristig erscheint der Weg über ein PPA sinnvoller.

5.5 Microsoft Windows

Der Quellcode für eine „PersoApp“-Software wird in der Programmiersprache Java geschrieben. Java wurde anfangs von der Firma Sun Microsystems entwickelt. Ab dem Jahr 2010 ist sie ein Tochterunternehmen der Oracle Corp. und wird dort weiterentwickelt. Eines der Vorteile der Programmiersprache ist die Unabhängigkeit von der zugrunde liegenden Plattform. Das bedeutet, ein einmal geschriebenes Java-Programm kann ohne Anpassungen von verschiedenen Betriebssystemen ausgeführt werden. Daher wurde die Sprache auch unter dem Slogan „Write once, run anywhere“ vermarktet.

Ein Java-Programm wird zunächst in Klartextform entwickelt. Nach Abschluss erfolgt die Übersetzung in Binärform. Bei vielen Programmen übernimmt nun das Betriebssystem bzw. der Prozessor die Aufgabe, das Programm auszuführen und die Anweisungen zu steuern. Java hat die Besonderheit, dass eine virtuelle Maschine (Java Virtual Machine, JVM) zwischengeschaltet ist. Diese JVM lädt den Binärcode und führt einige Prüfungen durch. Im Anschluss an die Prüfungen wird der Code durch die virtuelle Maschine ausgeführt. Die JVM übernimmt für den Entwickler einige Aufgaben, wie Speicherverwaltung, Typprüfung und anderes.

Die JVM wird von verschiedenen Quellen bereitgestellt. Derzeit sind insbesondere das Sun/Oracle JDK und das OpenJDK zu nennen. Früher pflegte Apple für Mac OS X ein eigenes JDK. Mittlerweile ist dies in das OpenJDK übernommen worden.

Die oben genannten Werkzeuge sind für eine Vielzahl von Betriebssystemen erhältlich. Dazu zählen insbesondere Microsoft Windows, GNU/Linux und Mac OS X.

5.5.1 Java-Archiv-Dateien

Alle Dateien, die für die korrekte Ausführung eines Programmes wichtig und notwendig sind, können innerhalb eines Java-Archivs zusammengeführt werden. Diese Dateien besitzen die Dateiendung „.jar“ und sind dem internen Format nach ZIP-Dateien. Java-Archive, oder kurz JAR-Dateien, dienen der Verteilung von Java-Programmen. In der Regel sind in einer JAR-Datei mehrere Java-Dateien in Binärform (Klassendateien) und eine Manifest-Datei enthalten.

Die Manifest-Datei trägt den Namen „MANIFEST.INF“ und liegt im Unterverzeichnis „META-INF“. Die Datei besteht aus Klartext und enthält Informationen über die enthaltenen Klassendateien. Eventuell finden sich in der Manifest-Datei auch Informationen über Java-Beans. Insbesondere kann der Name der Hauptklasse eines Java-Programms in der Datei festgehalten sein. Der genaue Aufbau kann der Spezifikation entnommen werden.[3]

Der besondere Vorteil in der Verwendung von JAR-Dateien liegt bei der direkten Ausführbarkeit: Sobald nachdem die Datei heruntergeladen wurde, kann die Software ohne Installation direkt ausgeführt werden. Damit entfällt der Schritt der Installa-

[3] http://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html

tion. Außer der JAR-Datei ist es nicht notwendig, weitere Dateien oder Installationsprogramme zur Verfügung zu stellen.

Der Nachteil der Lösung mittels JAR-Dateien besteht im Update. Es ist schwer, den Endnutzer über neue Versionen der Software zu informieren. Üblich ist es, dass diese Informationen über einen anderen Kanal (Webseite, Newsletter etc.) zur Verfügung gestellt werden.

5.5.2 Java Web Start

Die Technologie von Java Web Start beseitigt den Nachteil der fehlenden Updates. Es ist ein Verfahren, um unabhängig von der Plattform Software sicher und robust zu verteilen. Die ausführbaren Dateien lagern in dem Fall auf einem Webserver. Der Browser des Endanwenders verbindet sich mit dem Webserver und lädt die jeweils aktuelle Version der Software herunter.

Für Java Web Start muss seitens der Entwickler eine JNLP-Datei erzeugt werden. Das „Java Network Launching Protocol (JNLP)“ ist ein XML-Datenformat und enthält Angaben zu JAR-Dateien, zu Klassenpfaden und Parametern für das Programm. Die JNLP-Datei wird im Webserver hinterlegt. Gleichzeitig muss sichergestellt werden, dass der Webserver den korrekten MIME-Typ für die JNLP-Dateien ausliefert.[4]

Wenn der Endanwender die Software benutzen will, klickt er auf den Link im Browser. Die JNLP-Datei wird heruntergeladen und an die Java-Laufzeitumgebung weitergereicht. Die Laufzeitumgebung stellt den Pfad zu der JAR-Datei fest, lädt diese herunter und startet sie. Der wesentliche Unterschied besteht im Download der jeweils aktuellen Version der JAR-Datei.

5.5.3 Ausführbare Datei

Viele Programme für Microsoft Windows werden im PE-Format mit der Endung „.exe“ ausgeliefert und gestartet. Für Java-Programme gibt es ebenfalls die Möglichkeit, diese in als EXE-Dateien zu formatieren. Allerdings sind einige Einschränkungen zu beachten.

Die hauptsächliche Einschränkung liegt in der Verfügbarkeit entsprechender Werkzeuge. Im Open-Source-Umfeld existiert der GNU Compiler für Java (GCJ). Seit 2009 fand keine Weiterentwicklung statt. Es ist daher damit zu rechnen, dass dieser für künftigen Einsatz nicht geeignet ist. Ebenso ist die Software JSmooth[5] zu sehen. Dort liegt das letzte Update mehr als fünf Jahre zurück.

JExePack[6] ist ein Werkzeug für die Kommandozeile und erzeugt ausführbare Dateien für 32-Bit-Systeme. Diese Einschränkung erscheint für den zukünftigen Gebrauch zu stark. Daher ist auch die Software nicht zu empfehlen.

---

[4] Der korrekte MIME-Typ ist `application/x-java-jnlp-file`.

[5] http://jsmooth.sourceforge.net/

[6] http://www.duckware.com/jexepack/

Launch4J[7] packt Java-Programme in ausführbare Dateien ein. Das bedeutet, dass die EXE-Datei, welche von Launch4J erzeugt wird, nicht direkt von Microsoft Windows ausgeführt wird. Stattdessen werden die Java-Dateien wieder von der virtuellen Maschine ausgeführt. Dies bietet daher keinen wesentlichen Vorteil gegenüber den JAR-Dateien. Daher wäre für den Projektverlauf zu prüfen, ob Endanwender EXE-Dateien bevorzugen. Falls dem so ist, wäre Launch4J eine Alternative zur Erzeugung der ausführbaren Dateien.

Im kommerziellen Umfeld gibt es eine Reihe weiterer Programme. Einige davon erzeugen offensichtlich korrekt EXE-Dateien. Für das Open-Source-Projekt „PersoApp“ würde der Einsatz dieser Werkzeuge eine Einschränkung mit sich bringen. Denn so könnten nur diejenigen, die die Werkzeuge bezahlen können, diese auch nutzen. Daher ist von kommerziellen Werkzeugen eher abzuraten.

5.6 Mac OS X

Im Abschnitt 5.5 zu Microsoft Windows wurde ausführlich auf die Möglichkeiten von JAR-Dateien und Java Web Start eingegangen. Da dies unabhängig von der verwendeten Plattform zuverlässig funktioniert, soll auch für Mac OS X dieser Weg beschritten werden. Nutzer von Mac OS X können über die Webseite der „PersoApp“-Community[8] eine JAR-Datei herunterladen und diese lokal auf dem Rechner ausführen. Über die PersoApp-Webseite steht weiterhin eine URL zur Verfügung, über die die JNLP-Datei für Java Web Start geladen werden kann. Damit steht für das System eine einfach zu nutzende Form zu Verfügung.

6 Abläufe

Durch eine sorgfältige Prüfung der hier getroffenen Festlegungen wurde sichergestellt, dass diese Prozessbeschreibungen für die Tests und die Major Releases praktikabel und angemessen sind. Sollte sich dennoch im Projektverlauf die Notwendigkeit für eine Anpassung ergeben, dann ist dies einerseits an alle Projektteilnehmer explizit und verständlich zu kommunizieren, zum anderen gilt es die entsprechende Änderung detailliert festzuhalten und nachvollziehbar zu dokumentieren.

7 Mitgeltende Dokumente

- „D02-QM Qualitätskriterien: Aufbau, Messgrößen und Bewertung“
- „D06-QM Architekturkonzept der Open-Source-Core“

[7] http://launch4j.sourceforge.net/

[8] http://www.persoapp.de/