

KDE: Requirements, Processes and Architecture *

[An overview of the KDE Project]

Martin Root
st100506@stud.uni-stuttgart.de

Christian Bäumlisberger
st114381@stud.uni-stuttgart.de

Velihan Bulut
st103204@stud.uni-stuttgart.de

Stefan Schmid
st140748@stud.uni-Stuttgart.de

Stefan Schnack
st140187@stud.uni-stuttgart.de

ABSTRACT

Dieses Paper befasst sich mit dem Open Source Produkt KDE.

Es sollen sowohl die Architektur der Software als auch die Abläufe bei der Entwicklung genauer betrachtet werden. Um zu diesen Punkten zu gelangen, spielen natürlich erst einmal die Anforderungen eine Zentrale Rolle. Ohne diese könnte gar keine zielführende Entwicklung vorangetrieben werden. In der Gesamtheit soll hiermit ein Überblick über den Entwicklungsprozess hinter KDE gegeben werden. Aufgrund der Vielfalt und Menge ist es nicht möglich, restlos alle Teilbereiche zu betrachten.

Keywords

KDE, Requirements, Preosses, Architecture

1. INTRODUCTION

2. GRUNDLAGEN

In diesem Kapitel werden die Grundlagen dieser Arbeit beschrieben. Dazu gehören unter anderem einige technischen Details über KDE, Wissenswertes über die Versionshistorie, sowie der aktuelle Stand des Projekts.

2.1 Übersicht

In der heutigen Computerwelt ist die Interaktion zwischen Mensch und Computer ohne eine graphische Benutzeroberfläche nahezu undenkbar. Daher gibt es eine große Auswahl an unterschiedlichen Betriebssystemen, die auch unterschiedliche Benutzeroberflächen haben. Eines dieser grafischen Benutzeroberflächen ist KDE.

KDE stand ursprünglich für *Kool Desktop Environment* und steht heute für *K Desktop Environment*, zu deutsch *K Arbeitsumgebung*, wobei das *K* keine bestimmte Bedeutung mehr hat, sondern als eine Art Markenzeichen von KDE ist. Es handelt sich dabei um eine graphische Arbeitsumgebung für UNIX-Betriebssysteme. Es besteht aus einer Reihe von kleinen Programmen, einem Fenstermanager, einem Dateimanager und einigen Hilfsprogrammen. Das Ziel von KDE ist es, die Verwendung von UNIX-basierten Betriebssystemen zu vereinfachen. [10]. Bild 1 zeigt das aktuelle offizielle KDE Logo.

*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.



Figure 1: Aktuelles offizielles KDE Logo. [1]

2.2 Versionshistorie

Das KDE Projekt wurde am 14. Oktober 1996 von Matthias Ettrich begonnen. Sowohl der Name also auch der Funktionsumfang des Projekts orientierte sich an der proprietäre Desktop-Umgebung CDE (*Common Desktop Environment*). KDE setzte von Beginn an auf die Programmiersprache C++ und auf die umfangreiche Oberflächenbibliothek *Qt*.

2.2.1 KDE 1.x

Die KDE-Komponenten wurden ziemlich unkoordiniert entwickelt, weshalb es keine einheitlichen Alpha-Version gab. Etwa ein Jahr nach der Gründung von KDE erschien am 20. Oktober 1997 die erste Beta-Version. Nach drei weiteren Beta-Versionen wurde am 12. Juli 1998 die Version 1.0 von KDE präsentiert und veröffentlicht. Abbildung 2 zeigt einen Standbild der Benutzeroberfläche von K Desktop Environments Version 1.0. Trotz einiger Kritik wegen der unfreien Bibliothek Qt konnte sie sich durchsetzen und fand seinen weg in einige Linux-Distributionen. [?]

In den folgenden beiden Jahren wurde KDE durch zahlreiche Verbesserungen in einigen Punkten den Wünschen von Nutzern angepasst und auf die Version 1.1 gebracht. Durch eine neue Version der Oberflächenbibliothek Qt stand KDE vor vielen Verbesserungen, weshalb man sich entschied, die Verbesserungen nicht in die erste Generation von KDE aufzunehmen. Dieser Versionssprung wurde dazu genutzt die

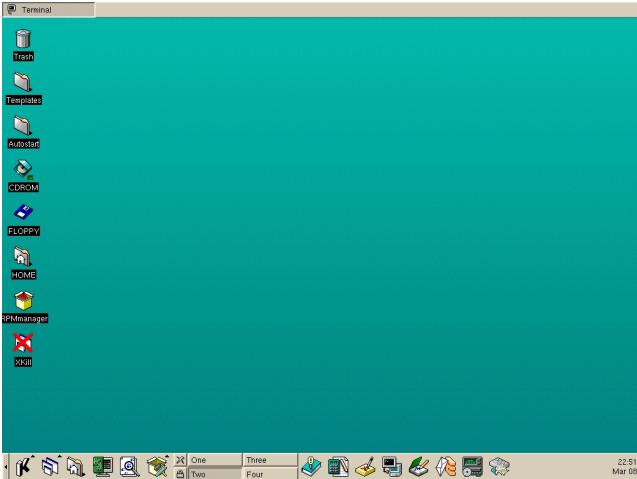


Figure 2: KDE Version 1.0 [2]

unkoordinierte Entwicklung und die Infrastruktur des Projekts zu überarbeiten. Bis zu Veröffentlichung wurde auch Qt unter die Lizenz GPL 2.0 gestellt, wodurch der Lizenzkonflikt zwischen den Lizenzen von KDE und Qt behoben wurde.

2.2.2 KDE 2.x

Die stabile Version von K Desktop Environments 2.0 wurde schließlich am 23. Oktober 2000 veröffentlicht. Abgesehen von den der besseren Infrastruktur wurde diese Version nun mit der freien Bibliothek Qt 2.2 veröffentlicht. Des Weiteren erntete auch *Konqueror*, der neue KDE-Dateimanager und -Webbrowser, viel positive Kritik. Andere Websbrowser waren zu dieser Zeit sehr unstabil oder nicht fertiggestellt. Abbildung 3 zeigt die Benutzeroberfläche von K Desktop Environments 2 und den Webbrowser Konqueror. Durch KDE 2.0 konnte sich das Unternehmen als feste Institution unter den X11-Oberflächen durchsetzen. [2, 3]

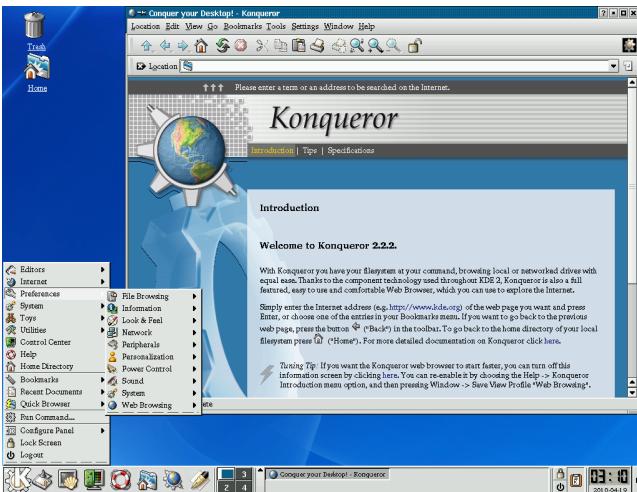


Figure 3: KDE Version 2.2.2 [2]

2.2.3 KDE 3.x

Nach zahlreichen Verbesserungen und zwei weiteren Veröffentlichungen von KDE in der zweiten Generation erschien am 3. April 2002 die Version 3.0. Diese und die darauffolgenden Versionen der 3. Generation brachten viele Neuerungen. Dazu zählten beispielsweise das *Desktop Sharing Framework*, wodurch KDE-Desktops von entfernten Rechner verwendet werden konnten, *Tabbed Browsing* in Konqueror und einen integrierten *Personal Information Manager* namens *Kontact*, das in einer Anwendung nützliche Funktionen wie E-Mail, Adressbuch, Kalender, Terminplaner, Newsreader, Wetteranzeige, Geburtstagserinnerung, Notizblock und Aufgabenliste kombinierte. Außerdem wurde der Benutzeroberfläche durch überarbeitete Icons und (in einigen Bereichen) durch neues Design ein modernes Aussehen verliehen. [2, 3] In Abbildung 4 ist K Desktop Environments in der Version 3.5 zu sehen.

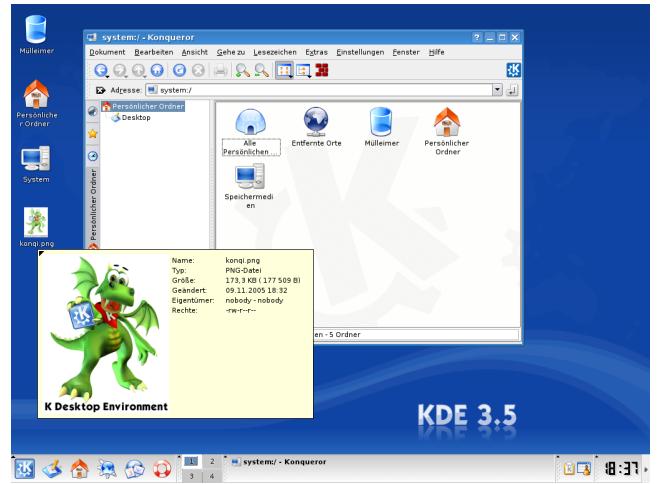


Figure 4: KDE Version 3.5 [2]

2.2.4 KDE Plasma Workspaces 4

In der darauffolgenden Version wurde die Ausformulierung K Desktop Environments nicht mehr verwendet und es erschien am 11. Januar 2008 KDE 4. Es wurde auf Basis von Qt 4 entwickelt. Viele Teile von KDE wurden von Grund auf neu programmiert, wie etwa der Desktop Plasma, also die Desktopumgebung und KDE. Zusammen mit KDE 4.2 wurden alle Funktionen aus KDE 3.x portiert und die Software war so stabil, dass die Nutzung von ihr erstmals auch für Endanwender empfohlen wurde. [3] Abbildung 5 zeigt die Benutzeroberfläche von KDE in der Version 4.2.

Zusammen mit KDE 4 begann die Community mit einem zeit-basierten Release-Plan. Feature-Releases wurden demnach jedes halbe Jahr und Bugfix-Releases jeden Monat veröffentlicht. Die halbjährlich veröffentlichten Softwarekomponenten waren fortan unter dem Namen *KDE Software Compilation (KDE SC)* bekannt. [3]

2.2.5 KDE Plasma 5

KDE Plasma ist die 5. Generation der KDE Arbeitsumgebung und stellt heute das Hauptprodukt der Community dar. Sie wurde am 15. Juli 2014 veröffentlicht. Durch den Wechsel auf Qt 5 wurde die Grafikausgabe auf Open GL umgestellt und bot viele neue visuelle Features. [4] Weitere Informationen zu KDE Plasma 5 werden in Abschnitt 5.2 be-

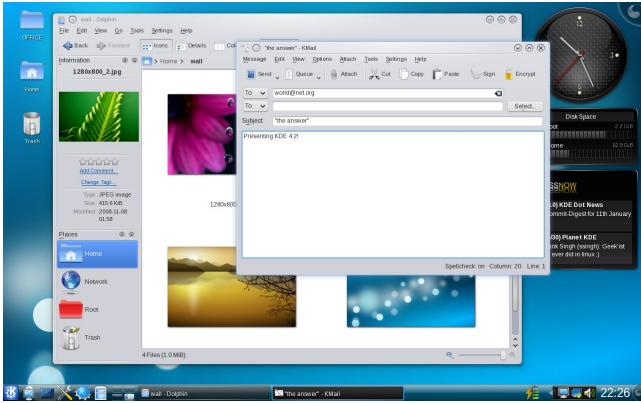


Figure 5: KDE Version 4.2 [3]

schrieben. Abbildung 6 zeigt ein Bild der Arbeitsumgebung.

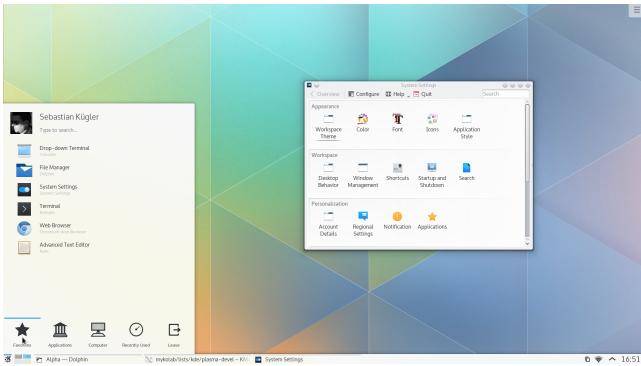


Figure 6: KDE Version 5.0 [4]

3. KDE REQUIREMENTS ENGINEERING

Der folgende Teil beschreibt das Requirements Engineering bei dem Open Source Projekt KDE. Ein Großteil der Informationen wurde der offiziellen deutschsprachigen Webseite von KDE entnommen[11].

3.1 Requirement Engineering

Requirements Engineering umfasst das Ermitteln, Analyseren, Spezifizieren und Validieren aller Eigenschaften und Rahmenbedingungen eines Softwaresystems, die über seinen gesamten Lebenszyklus gewünscht werden bzw. relevant sind[20].

3.2 Requirement Engineering in Open Source Projekten

Requirement Engineering bei Open Source Projekten unterscheidet sich von dem gewöhnlicher Projekte (wie z.B. bei Firmen die als Dienstleister Software entwickeln). Normalerweise werden die Anforderungen vom Kunden entgegen genommen und zusammen mit dem Ersteller der Software zu realistischen und aufgabenorientierten Requirements umformuliert. Doch bei Open Source Projekten gilt es bestimmte Ziele zu erfüllen, wie das Endprodukt ausehen soll, dann werden Requirements von der Community gegeben und von den Entwicklern erstellt.

3.3 Quellen für Requirements bei KDE

Das Sammeln von Requirements bei KDE erfolgt über Benutzer Feedback, Fehlerrückmeldungen und wirtschaftlichen Interessen. Feedback und Fehler werden auf der Webseite über eine Adresse angenommen[14].

3.4 Dokumentation von KDE

Die Dokumentation von KDE wird mit dem Produkt selbst ausgeliefert und dient auch als hilfe Datei. Deswegen ist sie mehr als Handbuch bzw. als Hilfestellung gedacht und weniger zum formulieren, festhalten oder entnehmen von Requirements. Als Beispiel wird einem gezeigt wie man die Lautstärke für Ausgabegeräte einstellt aber nicht welche Anforderungen an den Lautstärke Regler gestellt wurden[12].

3.5 Umgang mit Requirements bei KDE

Die Arbeit an KDE verläuft ähnlich zu der bei Linux. Eine Aufgabe wird in Teilaufgaben aufgeteilt, dann werden verantwortliche für diese Teilaufgaben gesucht und denen die Teilaufgabe übertragen. Wodurch diese als Ansprechpartner und Verantwortlicher dafür dienen. Als Beispiel kann man in folgendem Bild KDEgames sehen. Das Bild zeigt einen

kdegames

Status	Project	Description	Contact
DONE	KGoldrunner	Add a new game with 70 levels: Beginner's Luck, by Jason Self.	Ian Wadham <iandw.au@gmail.com>
DONE	KGoldrunner	Add a new game with 32 levels: Dig While Falling, by Jason Self.	Ian Wadham <iandw.au@gmail.com>
DONE	KGoldrunner	Add a menu action to save a recording of a level as a solution, using a file in the user's data area.	Ian Wadham <iandw.au@gmail.com>
DONE	KGoldrunner	Provide recorded solutions to the levels in the Tutorial game.	Ian Wadham <iandw.au@gmail.com>

Figure 7: KDEgames[?]

Ausschnitt einer Tabelle die der Quelle, der KDE Website entnommen wurde[?]. Die Tabelle hat vier Spalten. Die Erste zeigt den Status der Aufgabe an, DONE IN PROGRESS oder TODO. Die zweite zeigt den Namen des Projekts. Die Dritte eine Beschreibung des Projekts und die vierte den Verantwortlichen für das Projekt.

Dadurch wird die Bearbeitung der einzelnen Projekte dokumentiert.

Besprochen werden die Requirements und andere Aufgabenspezifische teile über Mailinglisten. Damit werden Diskussionen und Absprachen durchgeführt. Diese kann man ebenfalls über die Webseite einsehen, um einen Einblick in den Verlauf eines Projekts zu erhalten.

Zusätzlich werden noch Requirements an bestimmte Teile des Produkts wie z.B. Software die enthalten sein muss gestellt. Also um bei der KDE Auslieferung z.B. bestimmte Networking Funktionen zu unterstützen benötigen wir rdesktop. Das nachfolgende Bild zeigt ein Beispiel für Networking und Browsing. Das Bild zeigt zwei Tabellen die der Quelle, der KDE Website entnommen wurden[?]. Die Tabellen haben jeweils 5 Spalten. Die erste Spalte enthält den Namen des Packages das unter Umständen benötigt wird. Die zweite Spalte enthält das Level, also wie stark die Funktion abhängig von dem Requirement ist, unterteilt in Required, Recommended und Optional. Die dritte Spalte enthält eine Beschreibung was dieses Package enthält und die vierte Spalte eine Erklärung warum dieses Requirement für die

Networking

Package	Level	Description	Explanation	Modules
pppd	High	Pppd is a PPP (Point-to-Point Protocol) daemon. PPP is a common protocol for connecting to the Internet via a modem.	Kppp uses pppd to connect to the Internet using a computer POTS modem.	kdenetwork
LibVNCServer / LibVNCClient >= 0.9.1 (download)	Medium	LibVNCServer / LibVNCClient is a library for VNC support.	KRDC uses LibVNCClient for VNC client support. Krfb uses LibVNCServer for VNC server support.	kdenetwork
rdesktop >= 1.5.0	Medium	rdesktop is a Remote Desktop Protocol (RDP) client.	KRDC uses rdesktop for RDP support.	kdenetwork

Browsing

Package	Level	Description	Explanation	Modules
java >= 1.3	Recommended	Java is a programming language designed for use in the distributed environment of the Internet. Since Netscape integrated Java into its browser, Java has become a popular language for websites.	Some websites require the use of Java for some or all of their services. In addition, Java bindings exist for KDE which enables writing KDE applications in the Java language.	kdebase, kdebindings

Figure 8: Networking und Browsing[?]

Funktion nötig ist. In der fünften und letzten Spalte ist festgehalten zu welchem Modul das Package gehört.

3.6 Fazit

Abschließend kann ich zum Requirements Engineering bei KDE sagen, dass das Vorgehen mir schon zu Beginn meiner Nachforschung ähnlich zu dem vorkam, was ich über das Vorgehen bei der Arbeit an Linux selbst erfahren habe. Das Aufteilen der Aufgaben an verschiedene Personen die dann zuständig sind und stark dafür verantwortlich die Requirements zu fassen bzw. zu bestimmen. Dies hat sich auch beim Abschluss meiner Nachforschung bestätigt, weshalb ich dieses Vorgehen im obrigen Teil auch wiedergegeben habe. Dieses Vorgehen ist für gewöhnliche Projekte eher untypisch und unterschiedet sich auch von den klassischen Methoden die in der Vorlesung aufgeführt wurden. Wobei dazuzusagen ist, dass KDE ein Open Source Projekt ist und deswegen eine andere Vorgehensweise erfordert und es genauso seine Berechtigung hat, den KDE und Linux sind beide erfolgreiche Projekte.

4. PROZESSE

Dieses Kapitel befasst sich mit der Betrachtung der KDE Entwicklungs- und Organisationsprozesse. Nach der Beleuchtung des sogenannten Application Lifecycle, also wie Neuentwicklungen gehandhabt werden, betrachten wir das Projektmanagement, sowie die Kommunikation und Arbeit der verschiedenen Teams.

4.1 KDE Application Lifecycle [5]

In Abbildung 9 ist schön zu sehen, wie der Lebenszyklus einer KDE-Anwendung aussieht, welche auf Eigeninitiative eines Entwicklers implementiert wurde.

Zu Beginn kann lokal mit der KDE-Entwicklungsumgebung programmiert werden. Möchte man seine Anwendung in einem der kommenden Releases enthalten haben, muss man den entsprechenden Code samt Dokumentationspapier ins Subversion Repository committen (Wird momentan komplett auf Git umgestellt). Speziell dafür gibt es die sogenannte Spielwiese (/trunk/playground), auf welcher ohne jede Einschränkung programmiert werden darf.

Sollte sich der Entwickler dazu entscheiden, die neue Anwendung gerne zusammen mit dem Release Plan von KDE zu veröffentlichen, wird das Projekt auf die nächste Stufe geschoben, dem sogenannten KDE Review.

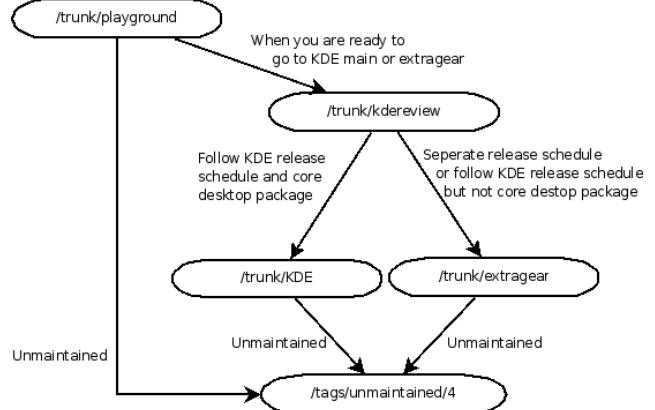


Figure 9: KDE Application Lifecycle [5]

Dieser Schritt stößt ein zweiwöchiges Review innerhalb der KDE Community an. Führte diese zweiwöchige Phase zu keinerlei Änderungswünschen, hat der Entwickler die Erlaubnis, seine Neuentwicklung ins produktive KDE zu übernehmen. Je nachdem, ob man dem festen Release Plan von KDE folgen oder darin frei sein möchte, wird das entsprechende Verzeichnis gewählt.

Eine letzte Hürde für den KDE Release Plan ist die Zustimmung des entsprechenden Modulkoordinators.

Sofern im Review-Status erst noch Änderungen und Bugfixes anstehen, darf je nach Verfügbarkeit des Entwicklers das Projekt bis zur Fehlerbehebung im Review-Status bleiben oder muss auf die Spielwiese zurückgesetzt werden. Dadurch befinden sich nur diejenigen Projekte im Review-Status, an welchen noch regelmäßig weiterentwickelt wird und keine Altlast-Projekte. [5]

4.2 KDE Release Team [17]

Das KDE Release Team ist zuständig für die Koordination aller offiziellen KDE Releases. Dies beinhaltet unter Anderem die Erstellung eines Zeitplans, die Einplanung von Marketingaufwänden, als auch die Kontrolle zur Einhaltung von Deadlines oder Code-Restriktionen.

Der Entscheidungsfindungsprozess soll laut offizieller KDE-Richtlinie öffentlich sein und es können Anregungen sowie Wünsche von allen Seiten eingereicht werden.

Personell setzt sich das Release Team aus den einzelnen Modulkoordinatoren der verschiedenen KDE-Module, sowie dem Marketing Team und diversen Personen für die Verwaltung der Planungen zusammen.

Die Kommunikation erfolgt wie üblich über Mailing-Listen oder direkte E-Mails. [17]

Im Auszug aus Abbildung 10 ist deutlich zu sehen, dass nicht einmal die Hälfte der Module bislang einen fixen Modulkordinator zugewiesen haben. Genau genommen existieren gerade einmal für 9 der 23 KDE-Module solche Personen.

Selbst vermeintlich wichtige Posten wie für die Systemadministration sind nicht offiziell besetzt, was sich als großer Schwachpunkt der aktuellen Projekt-Organisation herausstellen könnte.

Die Aufgabe des gesamten Release Teams besteht erstens darin, sicherzustellen, dass sich die Neuentwicklungen in ei-

Module	Description	Release Coordinator
kdelibs	KDE4-era foundational libraries	David Faure
KDE Frameworks	Qt5-based modern libraries	David Faure
kde-runtime	Applications required by KDE apps to function properly at runtime, such as a help browser, framework (e.g. phonon, solid) backends, and certain configuration modules	help wanted
kde-baseapps	Essential apps needed to complement a desktop shell for basic functionality (web browser, file manager, ...)	help wanted
kate	The KDE editor	Christoph Cullmann
konsole	The KDE Console	Kurt Hindenburg
Kdeaccessibility	Accessibility applications	help wanted
Kdeartwork kde-wallpapers kde-base-artwork	Additional icons, styles, etc.	help wanted
kdeadmin	Tools for system administration	help wanted
kdeedu	Applications with educational content	Aleix Pol Gonzalez
kdegames	Entertainment	help wanted
kdegraphics	Graphics viewing and editing	help wanted
kdemultimedia	Audio and video applications	help wanted

Figure 10: Auszug der Liste von KDE-Modulkoorinatoren [17]

nem auslieferbaren Zustand befinden.

Die zweite Kernkompetenz liegt darin, neue Features für zukünftige Releases zu sammeln und zu organisieren. Dabei entscheidet das Release Team auch, welche Features wirklich umgesetzt oder verworfen werden. [17]

Zusammenfassend lässt sich sagen, dass das Release Team eine zentrale Instanz für die Organisation der Projekte darstellt. Es beherbergt die Überwachung geplanter Entwicklungen, als auch die Entscheidungskraft für künftige Entwicklungen.

4.3 KDE Release Schedule - Beispiel Plasma 5 [18]

Dieses Kapitel zeigt einen beispielhaften Release Plan von Plasma 5 (siehe Abbildung 11). Der Zeitplan ist das Ergebnis des eben erläuterten Release Teams. Er schlüsselt chronologisch nacheinander die geplanten Versionen zusammen mit dem entsprechenden Release Datum und einer Kurzbeschreibung der enthaltenen Features auf. Ebenso werden die Daten für die entsprechenden Code Freezes vor größeren Releases definiert. Der Release Plan orientiert sich an der Planung des Release Teams und in diesem speziellen Fall auch dem Modulkoorinatoren von Plasma 5. Dieser plant gewöhnlich im Release Team die neuen Features wie in Abbildung 11. Alle enthaltenen Features müssen, wie schon erwähnt, zuvor im Release Team abgenommen werden. [18]

4.4 Code Reviews [16]

Für Code Reviews existieren KDE-seitig diverse Regeln, welche die Codequalität steigern sollen.

Thematisch betrachtet decken diese unter Anderem Sicherheitsaspekte, Wartbarkeit, Dokumentation, Stil ab.

Wie im KDE Application Lifecycle bereits verdeutlicht, müssen alle Codes im Review-Status, aber auch Bugfixes einem Review unterzogen werden. [16]

Version	Type	Date Tars	Date Release	Comments
	Repo Freeze	Thu 2015-11-05		Repo freeze, list of tars for release frozen today, no new repositories will be included in release after this.
5.4.95	Feature freeze and Beta	Thu 2015-11-19 (same day)	Tue 2015-11-19	Message and feature freeze, Plasma/5.5 branch made
5.5.0	Release	Thu 2015-12-03	Tue 2015-12-08	Final tag, only urgent fixes approved by release dude after tars please
5.5.1	Bugfix	Tue 2015-12-15	Tue 2015-12-15	Bug fixes and new translations only from Plasma/5.5 branch
5.5.2	Bugfix	Tue 2015-12-22	Tue 2015-12-22	Bug fixes and new translations only from Plasma/5.5 branch
5.5.3	Bugfix	Tue 2016-01-05	Tue 2016-01-05	Bug fixes and new translations only from Plasma/5.5 branch
5.5.4	Bugfix	Tue 2016-01-26	Tue 2016-01-26	Bug fixes and new translations only from Plasma/5.5 branch
5.5.5	Bugfix	Tue 2016-03-01	Tue 2016-03-01	Bug fixes and new translations only from Plasma/5.5 branch - only if needed
	5.6 Repo Freeze	Thu 2016-02-18		Repo freeze, list of tars for release frozen today, no new repositories will be included in release after this.
5.5.95	Feature freeze and Beta	Thu 2016-03-03 (same day)	Thu 2016-03-03	Message and feature freeze, Plasma/5.5 branch made
5.6.0	Release	Thu 2016-03-17	Tue 2016-03-22	Final tag, only urgent fixes approved by release dude after tars please
5.6.1	Bugfix	Tue 2016-03-29 (same day)	Tue 2016-03-29	Bug fixes and new translations only from Plasma/5.5 branch
		Tue 2016-04-05	Tue 2016-04-05	

Figure 11: Auszug des Release Schedule von KDE Plasma 5 [18]

5. KDE ARCHITECTURE

Im Folgenden wird nun die Architektur der vom Team des KDE Projekts entwickelten KDE Software Compilation (KDE SC) vorgestellt. Der wohl bekannteste Teil der KDE Software Compilation ist die Desktop Umgebung KDE Plasma. Dazu kommt eine große Anzahl an Anwendungen die KDE Applications. Diese werden vor allem in Verbindung mit KDE Plasma genutzt können aber auch unabhängig davon verwendet werden. Der dritte Bestandteil der KDE Software Compilation heißt KDE Frameworks und stellt eine Sammlung von Bibliotheken dar. Diese enthalten häufig benötigte Funktionen und machen es somit einfacher KDE Software zu entwickeln. Außerdem wird so das ständige neu entwickeln grundlegender Funktionen verhindert. Wichtig ist in diesem Zusammenhang auch die Qt Library die zwar nicht direkt zum KDE Projekt gehört aber eng mit dem KDE Projekt verbunden ist da Qt als Basis für die Entwicklung verwendet wird. Qt (<http://www.qt.io/>) ist im wesentlichen eine C++-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen die viele für die Entwicklung hilfreiche Funktionen bietet. Auf Anwenderebene sind also die KDE Applications und KDE Plasma die auf die KDE Frameworks und Qt aufbauen wie auch in Abbildung 12 dargestellt.

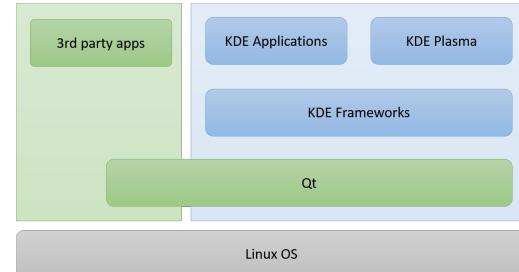


Figure 12: KDE Übersicht

5.1 KDE Frameworks 5

Das Ziel des unter der LGPL stehenden und vor allem in C++ und C entwickelten KDE Frameworks ist die Modu-

larisierung [13]. Aufbauend auf das als Basis genutzte Qt 5 stellt das aktuelle KDE Frameworks 5 grundlegende und für viele Anwendungsfälle nützliche Funktionen wie z.B. extra UI Elemente, Rechtschreibprüfung, usw. zu Verfügung. KDE Frameworks stellt somit durch seine verschiedenen Bibliotheken die Basis für KDE Plasma und KDE Applications dar und vereinfacht die Entwicklung, da auf erprobte Implementierungen zurückgriffen werden kann und unnötiges für jede Software neu entwickeln verhindert wird. Das KDE Frameworks 5 besteht dazu aus rund 60 einzelnen Bibliotheken mit verschiedenen Abhängigkeiten untereinander die möglichst Plattform unabhängig gehalten sind und versuchen so wenig wie möglich zusätzliche Abhängigkeiten zu verursachen [9]. Für eine bessere Übersicht welche Abhängigkeiten existieren erfolgt dabei eine Einteilung ist nach "Tier" und "Categorie" die im Folgenden auch noch detaillierter vorgestellt wird [8]. Abbildung 13 gibt einen groben Überblick für die Bestandteile der KDE Frameworks 5 und zu welchen "Tier" sie gehören.

Ziel der Aufteilung in viele einzelne in KDE Frameworks gebündelte Bibliotheken ist dabei dass es leichter möglich wird nur auf Teile der KDE Frameworks aufzubauen. Die vielen einzelnen Bibliotheken von denen immer nur die benötigen verwendet werden ist dabei eine noch recht neue für die Architektur der Software sehr wichtige Entwicklung die erst mit KDE Frameworks 5 im Dezember 2013 eingeführt wurde. KDE Frameworks in Version 4 trug noch den Namen KDE Platfrom baute noch auf Qt 4 und war im Prinzip nur eine einzige große KDElibs Library wie sie schon in den Versionen davor existierte.

Große umbauten auf KDE Frameworks Ebene wie z.B. von Version 4 auf 5 beeinflussen allerdings alle darauf aufbauende Software weshalb verschiedene KDE Frameworks Versionen parallel von verschiedenen Anwendungen genutzt werden können, so dass Anwendungen die darauf aufbauen in Ruhe beim erscheinen einer neuen Version auf diese portiert werden können. Major Releases (Versionsnummer X.0) brechen dabei die Abwärtskompatibilität und steigen auch auf die nächste große Qt Version um. Minor Veröffentlichungen (X.1, X.2, ...) hingegen garantieren Source und Binär Kompatibilität und sind meist kleine Weiterentwicklungen, Verbesserungen und vor allem Fehlerkorrekturen.

Bekannte Frameworks aus den KDE Frameworks 5 sind z.B. Sonnet eine Rechtschreib- und Grammatikprüfung, KHTML eine HTML und JS Library, Solid eine Hardware und Network Abstraktion und Phonon ein Multimedia Framework. Eine Vollständige Liste ist in der online API Dokumentation zu finden [13].

Die KDE Frameworks 5 haben dabei eine klare Dependency Struktur die in "Categorie" und "Tier" einteilt [8]. Dies hilft den Überblick zu behalten welche Abhängigkeiten der Einsatz eines bestimmten Framework mit sich bringt. In der online API Dokumentation ist deshalb auch eine Zuordnung der Frameworks zu den Tiers verfügbar.

Die Einteilung "Categories" bezieht sich dabei wie folgt auf Laufzeit-Abhängigkeiten [8]:

- **Functional:** Auf Qt aufbauend und ohne weitere Laufzeit-Abhängigkeiten z.B. KArchive, KPlotting, Threadweaver, KConfig, KCoreAddons
- **Integration:** Mit optionalen Laufzeit-Abhängigkeiten für Integration bzw. Kompatibilität mit Betriebssystem/Plattform darunter z.B. Sonnet, Solid

- **Solutions:** Haben gewollt Laufzeit-Abhängigkeiten um sich damit ergebende Vorteile nutzen zu können z.B. KIO, KService

Die Einteilung in "Tiers" bezieht sich auf die Compile-Zeit Abhängigkeiten. Von KDE Projekt wurden dazu die folgenden vier Tiers definiert [8]:

- **Tier 1:** Keine Abhängigkeiten innerhalb KDE Frameworks, nur Qt und andere kleine Abhänglichkeiten checker so dass eine einfach Verwendung in einem Qt Projekt möglich ist.
- **Tier 2:** Dürfen auf Tier 1 Frameworks aufbauen haben aber weiterhin einfach zu verwaltende Abhängigkeiten.
- **Tier 3:** Dürfen auf Tier 3 Farmworks genauso wie Tier 2 und Tier 1 Frameworks aufbauen und haben somit oft schon komplexere Abhängigkeiten.
- **Tier 4:** Für Anwendungsentwickler unwichtig vor allem Frameworks zur Integration.

Jedes Framework lässt sich somit klar einordnen. Außerdem gibt es eine vom KDE Projekt erstelle Tier/Categorie Matrix (siehe Abbildung 15) die verschieden sich damit ergebenden Kombinationen und Abhängigkeiten Zeigt.

5.2 KDE Plasma 5

KDE bietet zwei verschiedene Arbeitsflächen an. Jede ist dabei auf den Workflow und das entsprechend verwendete Gerät spezialisiert. So gibt es eine Arbeitsfläche für den normalen Schreibtisch-PC und Laptop-Computer und eine Arbeitsfläche für kleine Computer, wie beispielsweise Netbooks. Beide sind jeweils auf die entsprechenden Anforderungen angepasst worden. In Abbildung 16 und Abbildung 17 sind beide Arbeitsflächen zu sehen [19].

Seit KDE 4 wird die Arbeitsfläche in KDE Plasma genannt. Die neueste Version ist KDE Plasma 5 und stellt die bekannte für Linuxsysteme entwickelten Arbeitsplatzumgebung des KDE Projekts dar. Aufbauend auf Qt 5 und KDE Frameworks 5 findet die Entwicklung vor allem mit C++ und QML statt. Die Implementierung basiert auf dem Qt Graphics View Framework. Die benötigen Klassen befinden sich in der *libplasma* Bibliothek. Im Folgenden werden einige Klassen vorgestellt die eine zentrale Rolle bei der Entwicklung spielen [15].

- **Corona:** wird von *QGraphicsScene* abgeleitet und bietet die Funktionalität für das Hinzufügen von *Applets* und *Karamba Themes*
- **Widget:** wird von *QGraphicsItem* abgeleitet und funktioniert wie simple Elemente (zum Beispiel Icons oder auch Buttons) in der Arbeitsfläche
- **Applet:** wird von *Widget* abgeleitet und implementiert komplexere Funktionalitäten, wie beispielsweise eine Uhr oder Systemüberwachung. Ein Beispiel für die Uhr ist in Abbildung 16 zu sehen.
- **DataEngine:** die *DataEngine* dient dazu die Daten für ein Applet bereitzustellen, damit diese angezeigt werden können. Dies gilt für alle möglichen Daten.

KDE Plasma 5 ist eine moderne Arbeitsfläche, welche alle nötigen Werkzeuge bereitstellt um effizient und produktiv

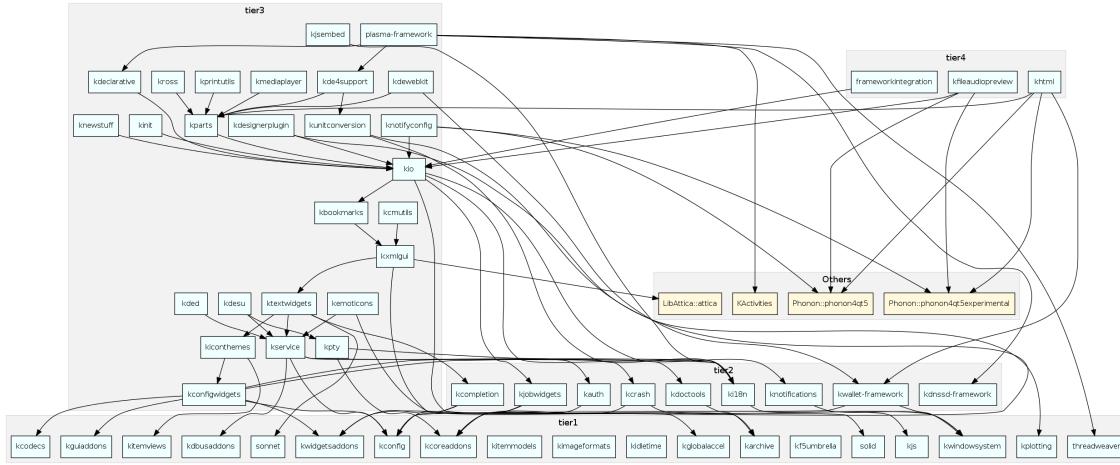


Figure 13: KDE Frameworks Übersicht [9]

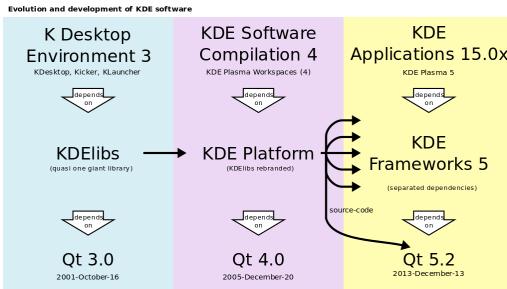


Figure 14: Entwicklungsgeschichte des KDE Frameworks [7]

arbeiten zu können. Weiterhin wird auch Wert auf das Aussehen gelegt. Mithilfe der neuen Technologien die verwendet werden, wird dieses Ziel auch erreicht. Dabei soll das Aussehen keinesfalls ablenken, sondern den Nutzer bei seiner Arbeit unterstützen. [15]

Weiterhin soll dem Nutzer die Möglichkeit gegeben werden den Desktop nach seinem Belieben zu gestalten. Dies wird durch Widgets ermöglicht, wovon tausende online zu Verfügung stehen. Auf Wunsch kann der Nutzer auch eigene Widgets entwickeln. Hierfür stehen Tutorials zur Verfügung.

5.3 KDE Applications 15.12

Viele einzelne Anwendungen die zum KDE Projekt gehören werden unter den KDE Applications zusammengefasst. Das KDE Projekt teils die über hundert Anwendungen deshalb zur besseren Übersicht in folgende Kategorien [6]:

- **Entwicklung:** dazu gehören zum Beispiel die Apps KDevelop, KLinkStatus und Umbrello.
- **Bildung:** dazu gehören zum Beispiel die Apps KTouch, Step und Marble.
- **Spiele:** dazu gehören zum Beispiel die Apps KMines, Kolf und KLines.
- **Graphik:** dazu gehören zum Beispiel die Apps KSsnapshot, Okular und KRuler.

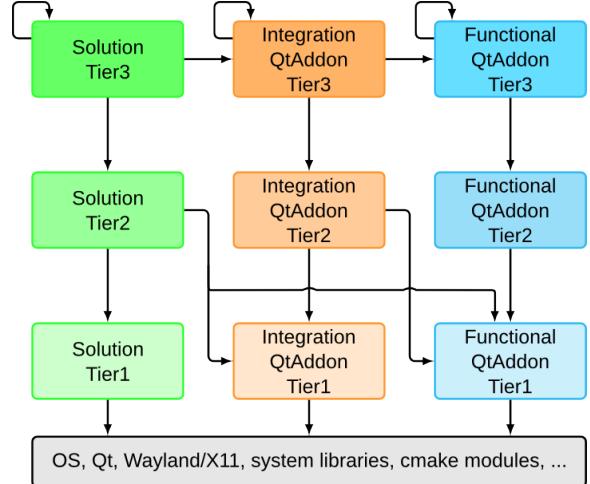


Figure 15: KDE Tier/Categorie Matrix [8]

- **Internet:** dazu gehören zum Beispiel die Apps Konqueror, KTorrent und KMail.
- **Multimedia:** dazu gehören zum Beispiel die Apps Amarok, KMix und KsCD.
- **Büro:** dazu gehören zum Beispiel die Apps Flow, Kontakt und Sheets.
- **System:** dazu gehören zum Beispiel die Apps Dolphin, KInfoCenter und KSystemLog.
- **Utilities:** dazu gehören zum Beispiel die Apps Ark, Kate und KCalc.

Die verschiedenen Anwendungen der KDE Applications bauen, wie KDE Plasma, auf Qt und KDE Frameworks auf. Davon Abgesehen sind die einzelnen Anwendungen aber relativ unabhängig vom gesamten Projekt gehalten, weshalb auch die Architektur der einzelnen Programme oft unterschiedlich ist.



Figure 16: KDE Plasma Desktop Arbeitsfläche [19]



Figure 17: KDE Plasma Netbook Arbeitsfläche [19]

6. CONCLUSIONS

Zusammenfassend kann gesagt werden, dass die Entwicklung eines Open Source Produkts eine komplexe Aufgabe darstellt. Die Entwickler sitzen nicht lokal in einer Firma, sondern verteilt über die gesamte Welt und kommunizieren vorwiegend über Mailing-Listen oder Kongresse.

Aufgrund dieser Herausforderungen existieren beim KDE Projekt idealerweise Verantwortliche und Leiter aller verschiedenen Module, welche unter Anderem die Softwarequalität sicherstellen sollen. Unterstützt werden sie von zahlreichen Guidelines (z.B. für Entwickler). Diese legen Richtlinien bezüglich Architektur, Stil, Oberfläche und vor allem auch Abnahmekriterien fest.

In der Praxis scheinen die verwendeten Prozesse recht ordentlich zu funktionieren. Regelmäßige Releases von KDE Plasma usw. bringen sowohl neue Features als auch viele Bugfixes. Die Verbreitung von KDE spricht auch für Kundenzufriedenheit.

Für die Zukunft muss das KDE Team jedoch weiter wachsen, denn einige Verantwortungen sind noch nicht fest vergeben. Dies erschwert es natürlich, weiterhin gute Software zu entwickeln und auch größere neue Features gut umzusetzen.

7. REFERENCES

- [1] KDE. <https://www.kde.org/stuff/clipart/klogo-official-oxygen-3000x3000.png>, 2015. [Online; accessed 25-December-2015].
- [2] KDE. https://de.wikipedia.org/wiki/K/Desktop_Environment, 2015. [Online; accessed 26-December-2015].
- [3] KDE. https://wiki.ubuntuusers.de/Geschichte_von_KDE/, 2015. [Online; accessed 26-December-2015].
- [4] KDE. <http://www.heise.de/open/artikel/Plasma-5-Der-KDE-Desktop-in-neuem-Glanz-2260451.html>, 2015. [Online; accessed 27-December-2015].
- [5] KDE. Application lifecycle. https://techbase.kde.org/Policies/Application_Lifecycle, 2015. [Online; accessed 28-December-2015].
- [6] KDE. Applications kde. <https://www.kde.org/applications/>, 2015. [Online; accessed 22-December-2015].
- [7] KDE. Evolution and development of kde software. https://en.wikipedia.org/wiki/File:Evolution_and_development_of_KDE_software.svg, 2015. [Online; accessed 27-December-2015].
- [8] KDE. Frameworks 5. <https://dot.kde.org/2013/09/25/frameworks-5>, 2015. [Online; accessed 27-December-2015].
- [9] KDE. Frameworks 5 tech preview. <https://dot.kde.org/2014/01/07/frameworks-5-tech-preview>, 2015. [Online; accessed 27-December-2015].
- [10] KDE. Häufig gestellte fragen über kde. <https://www.tu-chemnitz.de/urz/archiv/kursunterlagen/linux-bedienung/KDE3/ORIGINALE/userguide/frequently-asked-questions.html>, 2015. [Online; accessed 24-December-2015].
- [11] KDE. Kde. <https://de.kde.org>, 2015. [Online; accessed 27-December-2015].
- [12] KDE. Kde. <https://docs.kde.org/index.php?language=de&package=kdemultimedia>, 2015. [Online; accessed 28-December-2015].
- [13] KDE. The kde development platform. <https://www.kde.org/developerplatform/>, 2015. [Online; accessed 27-December-2015].
- [14] KDE. Kde techbase. https://techbase.kde.org/Development/Software\Engineering_\Framework\#Requirements_Gathering, 2015. [Online; accessed 28-December-2015].
- [15] KDE. Plasma kde. <https://techbase.kde.org/Development/Architecture/KDE4/Plasma>, 2015. [Online; accessed 22-December-2015].
- [16] KDE. Policies – suggested review criteria. https://techbase.kde.org/Policies/Suggested_Review_Criteria, 2015. [Online; accessed 28-December-2015].
- [17] KDE. Projects – release team. https://techbase.kde.org/Projects/Release_Team, 2015. [Online; accessed 28-December-2015].
- [18] KDE. Schedules – plasma 5. https://techbase.kde.org/Schedules/Plasma_5, 2015. [Online; accessed 28-December-2015].
- [19] KDE. Workspaces kde. <https://www.kde.org/workspaces/>, 2015. [Online; accessed 22-December-2015].
- [20] J. D. Susanne Patig. Requirements engineering. <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/Requirements-Engineering/index.html>, 2015. [Online; accessed 27-December-2015].