

Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Institut für Theoretische und Technische Informatik

Fachgebiet für Softwaresysteme / Prozessinformatik

Analyse und Entwicklung der webbasierten Konfiguration einer Software Systemfamilie als Linux-Live-CD im Rahmen des Digitalen Video Projektes

Diplomarbeit zur Erlangung des akademischen Grades
„Diplom-Wirtschaftsinformatikerin“

Vorgelegt an der Fakultät für Informatik und Automatisierung der
Technischen Universität Ilmenau von

Katharina Berg

Geboren am 06. Februar 1981 in Zeulenroda

Matrikel 99

Matrikelnummer 29709

Betreuende Hochschullehrerin: Prof. Dr. -Ing. habil. Ilka Phillipow

Betreuer: Dr.-Ing. Detlef Streitferdt

Beginn der Diplomarbeit: 01. Februar 2005

Abgabe der Diplomarbeit: 01. August 2005

Inhaltsverzeichnis

Abbildungsverzeichnis **5**

Tabellenverzeichnis **7**

Abkürzungsverzeichnis **9**

Danksagung **13**

Kurzfassung **15**

1 Einleitung **17**
1.1 Gegenstand der Arbeit und Problemstellung 17
1.2 Zielsetzung 18
1.3 Vorgehensweise und Aufbau der Arbeit 18

2 Grundlagen und Einführung **19**
2.1 Das DVP-Projekt an der TU Ilmenau 19
2.1.1 Vorstellung des Projektes 19
2.1.2 Das FORE-Modell 20
2.2 Live-CD Projekte 23
2.2.1 Bestehende Live-CD Projekte unter Linux 26
2.2.1.1 Knoppix 3.9 26
2.2.1.2 Kanotix 2005-03 27
2.2.2 Live-CD Projekte unter Windows 28
2.2.2.1 Microsoft Windows PE 28
2.2.2.2 Bart PE 28
2.3 Digitalfernsehen 29
2.3.1 Digitalfernsehen im Wohnzimmer 30
2.3.2 Digitalfernsehen am PC 31
2.4 Software-Projekte für digitales Fernsehen 32
2.4.1 Linuxbasierende Projekte 32
2.4.1.1 Das VDR-Projekt 32
2.4.2 Windowsbasierende Projekte 35

3 Konfigurationsmöglichkeiten von Software **37**
3.1 Allgemeine Einführung 37
3.1.1 Ablauf des Softwareentwicklungsprozesses 39

3.1.2 Konzepte der Systemfamilienentwicklung ... 40
3.1.3 Integration der Konzepte in den Softwareentwicklungsprozess ... 41
3.2 Softwareengineeringkonzepte zur Konfiguration von Systemfamilien ... 44
3.2.1 Einleitung in die Entwicklung von Systemfamilien ... 45
3.2.2 Programmierparadigmen ... 47
3.2.2.1 Generative Programmierung ... 47
3.2.2.2 Aspektorientierte Programmierung ... 50
3.2.3 Schichtenarchitekturen ... 52
3.2.3.1 GenVoca ... 52
3.2.3.2 Plugins ... 54
3.2.4 Programmierkonzepte ... 56
3.2.4.1 Entwurfsmuster ... 56
3.2.4.2 Polymorphismus ... 59
3.2.5 Komponentenarchitekturen ... 61
3.2.5.1 COM / ActiveX von Microsoft ... 61
3.2.5.2 JavaBeans von Sun ... 63
3.2.5.3 CORBA von OMG ... 65
3.3 Konfiguration im laufenden Betriebssystem ... 66
3.3.1 Windows ... 66
3.3.1.1 Beschreibung der Konzepte ... 66
3.3.1.2 Bewertung der Konzepte ... 70
3.3.2 Linux ... 72
3.3.2.1 Beschreibung der Konzepte ... 72
3.3.2.2 Bewertung der Konzepte ... 78
3.4 Ergebnisse der Analyse von Konfigurationskonzepten ... 79
3.4.1 Zusammenfassung der Ergebnisse ... 79
3.4.2 Ergebnisse für die Konfiguration der VDR-Software ... 83

4 Beispiel im Rahmen des DVP-Projektes ... 85
4.1 Ist-Analyse und Domänenwissen ... 87
4.1.1 Beschreibung der CD ... 87
4.1.2 Eingriffsbereiche ... 91
4.1.2.1 Bootvorgang ... 91
4.1.2.2 Die Weboberfläche ... 91
4.1.2.3 Der Programmcode ... 92
4.1.2.4 Integration neuer Plugins ... 93
4.2 Soll-Konzept: Mögliche Änderungen ... 94
4.2.1 Änderungen auf Code-Ebene ... 94

4.2.2 Änderungen auf Seiten des Webmenüs ... 96
4.3 Zusammenfassung der Ergebnisse ... 98

5 Zusammenfassung und Ausblick ... 99
5.1 Zusammenfassung und abschließende Bemerkungen ... 99
5.2 Ausblick ... 100

Anhang ... 103
Anhang A: Aufbau des DVP-Projektes an der TU Ilmenau ... 103
Anhang B: Merkmalmodell des DVP nach FORE ... 105
Anhang C: Plugins für VDR (Stand 01.07.2005) ... 107

Thesen ... 115

Literaturverzeichnis ... 117

Erklärung ... 125

Abbildungsverzeichnis

Abbildung 1: Phasen des Softwareentwicklungsprozesses 38
Abbildung 2: Konfigurationsmöglichkeiten im Laufe des Entwicklungsprozesses 43
Abbildung 3: Das Entwurfsprinzip der Systemfamilie 46
Abbildung 4: Systemfamilienentwicklung nach Czarnecki 47
Abbildung 5: Vorgehensweise der Aspektorientieren Programmierung 51
Abbildung 6: GenVoca-Schichten im Beispiel 53
Abbildung 7: GenVoca Grammatik im Beispiel 53
Abbildung 8: Aufbau einer Pluginarchitektur 55
Abbildung 9: Struktur eines Adapters 57
Abbildung 10: Schichtenmodell der komponentenbasierten Software 61
Abbildung 11: OMA Architektur von CORBA 65
Abbildung 12: Der Registrierungs-Editor 69
Abbildung 13: YaST 2 - die Systemsteuerung für Suse Linux 73
Abbildung 14: Konfiguration der VDR-Kanäle 74
Abbildung 15: Ansicht des KDE-Kontrollzentrums unter Suse Linux 9.2 76
Abbildung 16: Die dazugehörige Konfigurationsdatei für die Kontrollleiste 77
Abbildung 17: Eine XML-Konfigurationsdatei für Openoffice.org 78
Abbildung 18: Aufbau der Pluginarchitektur im Beispiel 86
Abbildung 19: Ansicht des Menüs der bestehenden CD 89
Abbildung 20: Vorgehensweise der Demo-CD 89
Abbildung 21: Umsetzung der Anforderungen als FORE-Notation 90
Abbildung 22: Zusammenspiel von Apache, Python und dem Menü 92
Abbildung 23: Prozedur- und Programmaufrufe im Hauptprogramm 92

Tabellenverzeichnis

Tabelle 1: Definition einer Live-CD 23
Tabelle 2: Einsatzmöglichkeiten von Live-CDs 24
Tabelle 3: Bewertung von Live-CDs 25
Tabelle 4: Knoppix Derivate 27
Tabelle 5: Vorteile des digitalen Fernsehens 29
Tabelle 6: Empfang des Digitalfernsehens 30
Tabelle 7: Das Leistungsspektrum der VDR-Software 33
Tabelle 8: VDR Distributionen 33
Tabelle 9: Software für Digitalfernsehen unter Windows 35
Tabelle 10: Definition von Konfiguration 37
Tabelle 11: Definition von Konfigurationsmanagement 37
Tabelle 12: Definition einer Komponente 45
Tabelle 13: Definition von Systemfamilien 46
Tabelle 14: Bewertung der Generativen Programmierung 49
Tabelle 15: Bewertung der Aspektorientierten Programmierung 51
Tabelle 16: Bewertung des GenVoca-Modells 54
Tabelle 17: Bewertung von Pluginarchitekturen 56
Tabelle 18: Bewertung von Entwurfsmustern 58
Tabelle 19: Bewertung des Polymorphismus 60
Tabelle 20: Bewertung der COM-Architektur 63
Tabelle 21: Bewertung der JavaBeans 64
Tabelle 22: Bewertung von CORBA 66
Tabelle 23: Bewertung der INI-Dateien / Textbasierender Konfigurationen 67
Tabelle 24: Aufbau der Registry unter Windows XP 68
Tabelle 25: Bewertung des Datenbank-Konzeptes 71
Tabelle 26: Bewertung linuxbasierter Konfigurationskonzepte 79
Tabelle 27: Gesamtbewertung 81
Tabelle 28: Plugins der Demo-CD 88
Tabelle 29: Pfade der Live-CD 90
Tabelle 30: Änderungen des Programm-Codes 94
Tabelle 31: Änderungen der Weboberfläche zur Erhöhung der Ergonomie 97

Abkürzungsverzeichnis

AFROS	Atari FRee Operating System
AOP	Aspektorientierte Programmierung
API	Application Programming Interface
ATI	Array Technology Inc.
AVI	Audio Video Interlaced
BeOS	Be Operating System
BIOS	Basic Input Output System
CORBA	Common Object Request Broker Architecture
CD	Compact Disk
CDDB	CD-Database
COM	Component Object Model
CPU	Central Processor Unit
CSS	Cascading Style Sheet
DCOM	Distributed Component Object Model
DIVx	Digital Video Express
DII	Dynamic Invocation Interface
DIN ISO	Deutsche Industrie Norm / International Standard Organisation
DLL	Dynamic Link Library
DSL	Domain Specific Language
DTD	Document Type Definition
DVB	Digital Video Broadcasting
DVB-C	Digital Video Broadcasting Cable
DVB-S	Digital Video Broadcasting Satellite
DVB-T	Digital Video Broadcasting Terrestrial
DVD	Digital Versatile Disk
DVDRAM	Digital Versatile Disk -Random Access Memory
DVI	Digital Visual Interface
DVP	Digital Video Project
EDU	Education

IESE	Fraunhofer Institut für Experimentelles Software Engineering
EJB	Enterprise Java Beans
EPG	Electronic Program Guide
EXE	Executable
FAQ	Frequently Asked Questions
FAST	Family-Oriented Abstraction, Specification, and Translation.
Fat32	File Allocation Table 32 bit
FH	Fachhochschule
FORE	Family-Oriented Requirements Engineering
GARV	Gesellschaft zur Förderung der Rundfunkversorgung
GB	Gigabyte
GNOME	GNU Network Object Model Environment
GNU	Gnome Non Unix
GP	Generative Programmierung
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HTML	Hypertext Mark-up Language
HTPC	Home Theatre Personal Computer
IBM	International Business Machines
IDL	Interface Definition Language
IDTV	Integrated Digital Television
IO	Input Output
INI	Initialization
IrDA	Infrared Data Association
IRQ	Interrupt Request
KB	Kilobyte
KDE	Kool Desktop Environment
LAN	Local Area Network
LCD	Liquid Crystal Display
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentification Code
MB	Megabyte
MHz	Megahertz

MP3s	Moving Picture Expert Group 1.0 Layer 3
MPEG	Moving Picture Expert Group
MS-DOS	Microsoft Disk Operating System
MTS	Middleware Transaction Server
NTFS	New Technology File System
OCX	OLE Control Extension
OLE	Object Linking and Embedding
OMG	Object Management Group
OO	Objektorientierung wegdamit
OOP	Objektorientierte Programmierung weg dmait
OMA	Object Management Architecture
ORB	Object Request Broker
OSD	On-Screen Display
PC	Personal Computer
PDA	Personal Digital Assistant
PE	Preinstallation Environment
PVR	Personal Video Recorder
RAM	Random Access Memory
ReactOS	Eigenname bzw. Reacting operation sytem schätzungsweise
RPM	RedHat Package Manager
SVCD	Super Video Compact Disk
TU	Technische Universität
TV	Television
TVOON	Eigenname
UML	Unified Modeling Language
USB	Universal Serial Bus
VCD	Video Compact Disk
VDR	Video Disk Recorder
WAV	Wave
WLAN	Wireless Local Area Network
XML	Extensible Markup Language
XP	Experience
YaST	Yet another Setup Tool

Danksagung

Als erstes möchte ich meinem Betreuer Herrn Dr.-Ing. D. Streitferdt danken, der mich in fachlicher und organisatorischer Hinsicht bei der Realisierung meiner Diplomarbeit unterstützt hat.

Ein großes Dankeschön gilt auch meiner Familie, die mir in der Zeit der Diplomarbeit stets den Rücken gestärkt hat und immer Lob und konstruktive Kritik parat hatte.

Des Weiteren danke ich meinen Freunden, die mich während der Arbeit unterstützt haben, immer ein offenes Ohr für mich hatten und die ich um Ratschläge in allen Angelegenheiten befragen durfte. Danken möchte ich vor allem Carina Viertbauer für die Bereitstellung ihres Laptops.

Und zum guten Schluss danke ich ganz besonders meinem ehemaligen Arbeitgeber, der Ergo Forschungsgesellschaft mbH in Hamburg, für die verständnisvolle Unterstützung und die Bereitstellung der Räumlichkeiten, in der ich meine Arbeit in produktiver Atmosphäre schreiben konnte.

Kurzfassung

Der moderne Mensch stellt **komplexe Anforderungen** quantitativer und qualitativer Art an die Technik der multimedialen Welt. Es werden mehr Funktionen zu besseren Preisen in einem ansprechenden Design, so aktuell wie möglich und einer Vielzahl von Schnittstellen für andere Geräte gefordert. Der Wandel vom reinen Nutzgerät zum Lifestyle-Produkt ist längst vollzogen. Im 21. Jahrhundert wird von Software erwartet, dass sie individuell maßgeschneidert ist, die stetig steigenden Kundenanforderungen möglichst komplett auf hohem Leistungsniveau abdeckt, und trotzdem aktuell und preisgünstig ist.

Die **Medien** durchdringen alle Bereiche des Lebens. Nachrichten, Unterhaltung, Kommunikation sind heute in allen Situationen nicht mehr wegzudenken. Die **Qualitätsansprüche** im Bereich des Fernsehens steigen, Digitalfernsehen erobert Deutschland und neue Endgeräte den Markt. So werden zu Hause in zunehmendem Maße digitale Videorecorder genutzt, die Aufnahmen auf DVDs brennen können. Nicht nur für den standardmäßigen Hausgebrauch, auch zukunftsweisende Angebote wie tragbare Video- und Audiorecorder mit Linux, WLAN und PDA-Funktionen werden auf Messen und Shows präsentiert.[1] Auch PCs, heute vollwertige **Multimedia-Anlagen**, werden zur Aufnahme von Fernsehsendungen verwendet. Viele Nutzer sind heute sicher im Umgang mit ihrem PC und nutzen ihn vielfältiger.

Im Bereich **Opensource-Software** gibt es interessante Projekte, die komplexe Software für **digitale Videorecorder** anbieten. Das Betriebssystem Linux ist für den alltäglichen PC-Nutzer eine Alternative geworden. Gerade **Live-CDs** wie Knoppix bieten die Möglichkeit, Linux einfach auszuprobieren. Letztendlich kommt der Opensource-Gedanke, ein Teil einer innovativen Gemeinschaft zu sein und Wissen offen zu teilen und zu vermehren, bei immer mehr Menschen an. Ein Beispiel ist der Erfolg der freien Online-Enzyklopädie Wikipedia, die 2001 ins Leben gerufen wurde und nun rund 236200 Artikel enthält.[2]

Aufgrund all dieser Entwicklungen gilt es, die Bedürfnisse nach individuell angepasster kostengünstiger Software insbesondere für Medien zu befriedigen. Grundlage hierfür sind **Softwarearchitekturen** und moderne Vorgehensweisen der Programmierung, die sich den Marktbedürfnissen angepasst haben und die **Entwicklung von Systemfamilien** fokussieren.

[1] Z.B. Von Archos, vgl. [Heis 2005] Newstickermeldung 54882
[2] Vgl. [Wiki 2005] Startseite

1 Einleitung

1.1 Gegenstand der Arbeit und Problemstellung

In der Kommunikationsgesellschaft verfügt eine Großzahl an Haushalten über einen gut ausgestatteten Komplett-PC mitsamt umfassender Medienausstattung wie TV-Karten, schnellen, leistungsfähigen Grafikkarten und Dolby Surround 5.1 (oder höheren) Anlagen. Es liegt nahe, die Vorzüge dieser Geräte auch im Medienbereich wie z.B. dem Digitalfernsehen stärker zu nutzen, da die Grenzen zur Unterhaltungselektronik immer mehr verschwinden. Solche Multimediaanlagen fristen ihr Dasein oft nicht mehr im Heimbüro, sondern sind der Mittelpunkt des modernen Wohnzimmers.

Standardsoftware zeigt einige Mängel auf, die der Bevölkerung durch die Medien bewusster geworden sind. Es wird leistungsfähigere Software gewünscht; mancher Nutzer möchte Software aktiv verbessern. Das Interesse an Opensource-Software nimmt so zu und ist nicht mehr nur ein Interessengebiet einer kleinen Gruppe. Die Vorteile liegen auf der Hand: Opensource-Software kostet oft nichts, ist durch die Verbreitung und die Weiterentwicklung immer auf dem aktuellsten Stand und lädt zum Ausprobieren ein: Im Internet finden Interessierte eine Fülle an Informationen.

Auf der anderen Seite gibt sich der Kunde mit dem Leistungspaket von Software oft nicht zufrieden. Moderne Ansprüche verlangen nach persönlich zugeschnittener Software. Eine Lösung hierfür ist das Konzept[3] der Systemfamilien. Ein Software-Kern, der die Grundfunktionen enthält, wird modular an die Ansprüche angepasst und so um diverse Funktionen erweitert. Ein Beispiel hierfür ist die Vielzahl an Steuererklärungssoftware für unterschiedliche Nutzergruppen. Das spart nicht nur Zeit im Angesicht steigenden Leistungsdrucks kürzer werdender Produktlebenszyklen, da die Module kontinuierlich wieder verwendet werden können und modular weiterentwickelt werden, sondern auch auf wirtschaftlicher Seite spart die konsequente Wiederverwendung Entwicklungskosten und dient vor allem der Kundenzufriedenheit.

Diese Trends sind aufgrund ihrer Entwicklung zukunftsweisend für die Softwareherstellung. Deswegen sollen sie verbunden miteinander im Mittelpunkt dieser Diplomarbeit stehen.

[3] Definition eines Konzeptes nach [Balz 2001] S. 38 und 44: Ein Konzept modelliert einen definierten Sachverhalt unter mehreren Gesichtspunkten. Zur Beschreibung eines Konzeptes nutzt man eine Notation.

1.2 Zielsetzung

Diese Arbeit konzentriert sich auf das Zusammenführen moderner Softwareanforderungen mit den Vorteilen von Digitalfernsehen. Dazu wird in die entsprechenden Domänen von Live-CDs, Digitalfernsehen und Systemfamilien eingeführt.

Hauptaufgabe ist es, Konzepte der Systemfamilienentwicklung entlang des Softwareentwicklungsprozesses zu analysieren und zu bewerten. Dafür werden verschiedene repräsentative Konzepte systematisch untersucht und ihre Vor- und Nachteile beschrieben. Als Beispiel wird dazu die Konfiguration einer kundenspezifischen Live-CD mit der Opensource-Software VDR und einigen Plugins vorgestellt, analysiert und Änderungen im Sinne des Software Reengineerings vorgeschlagen. Diese Demo-CD wurde in einer Studienarbeit von Melanie Schöppe und Martin Rulsch[4] prototypisch geschaffen; sie bietet ein webbasiertes Menü zur Auswahl spezifischer Plugins und startet die nach Kundenanforderungen konfigurierte VDR-Software. Diese Live-CD lässt sich im Rechnerlabor der TU Ilmenau booten und ausführen.

Für das praktische Beispiel wurde das an der TU Ilmenau von Herrn Dr.-Ing. Detlef Streitferdt entwickelte FORE -Modell zur Strukturierung von Kundenanforderungen genutzt.

1.3 Vorgehensweise und Aufbau der Arbeit

Nach den einleitenden Worten über diese Arbeit im **1. Kapitel** erfolgt im **2. Kapitel** eine Einführung in das DVP-Projekt und das FORE-Modell[5]. Es werden Live-CD Projekte und ihre Einsatzmöglichkeiten vorgestellt. Es wird auf den Empfang von Digitalfernsehen und Videorecorder am PC sowie ihre dazugehörige Software unter Linux und Windows eingegangen. Im **3. Kapitel** wird die Konfiguration von Software untersucht. Es werden verschiedene Betriebssysteme, Programmierparadigmen und Werkzeuge im Laufe des beschriebenen Softwareentwicklungsprozesses analysiert. Die Systematik besteht aus der Beschreibung des Konzeptes und der anschließenden Bewertung. Aus diesen Ergebnissen werden Schlüsse für die Konfigurationsdateien der VDR-Software auf der Live-CD gezogen. Im **4. Kapitel** wird die bereits existierende Live-CD vorgestellt und ein theoretisches Konzept zur Neu-Konfiguration anhand der Ergebnisse aus Kapitel 3 vorgeschlagen. Zuletzt erfolgt im **5. Kapitel** die Zusammenfassung der Erläuterungen und der Ergebnisse dieser Diplomarbeit. Abschließend zeigt der Ausblick, wohin die Entwicklung von Systemfamilien führen wird.

Im Anhang werden eine Reihe nützlicher Informationen zur Verfügung gestellt.

---

[4] Vgl. [Schö 2005]

[5] Vgl. [Stre 2005] S. 87 ff

2 Grundlagen und Einführung

Dieses Kapitel soll den Leser in die Themenbereiche dieser Diplomarbeit einführen. So wird als erster wichtiger Punkt erläutert, was Inhalt, Bedeutung und Ziel des DVP-Projektes an der TU Ilmenau sind. Dazu wird das FORE-Modell zur Erhebung von Anforderungen an Systemfamilien erläutert. Anschließend werden Live-CD Projekte vorgestellt. Ausgehend von der Entwicklung des digitalen Fernsehens und der entsprechenden Lösungen der Unterhaltungsindustrie wird die Bedeutung der softwaretechnischen Lösungen am PC deutlich. Hier wird auf Softwarelösungen von Microsoft Windows und Linux eingegangen. Zu diesem Zweck werden praktikable Live-CD Projekte vorgestellt.

2.1 Das DVP-Projekt an der TU Ilmenau

2.1.1 Vorstellung des Projektes

Das DVP-Projekt wurde 2002 von Herrn Dr.-Ing. Detlef Streitferdt an der Fakultät für Softwaresysteme / Prozessinformatik ins Leben gerufen, um unter anderem das in der Dissertationsschrift „Family-Oriented Requirements Engineering“[6] entwickelte FORE-Modell zu testen und zu evaluieren.[7] Der Leitgedanke des DVP besteht in dem Wunsch, die Möglichkeiten von Fernseher, PC und Audio-Geräten zu einem multimedialen Center zu vereinen.

In der Dissertation wurden ein Konzept und eine Notation geschaffen, um Systemfamilien zu beschreiben, die aus einem Systemkern bestehen und sich dazu optional je nach Kundenwunsch erweitern lassen. Diese Sichtweise der Softwareentwicklung hat enormes Entwicklungspotential, da sie Qualitäts-, Preis- und Zeitoptimierung anstrebt. Die Herangehensweise eröffnet die Möglichkeit, kostengünstig Individualsoftware aus bereits verwendeten Komponenten zusammenstellen zu können. Der Kunde wird dadurch bewusster in den Entwicklungsprozess einbezogen und die Kundenzufriedenheit steigt. Zur Evaluierung der Methodik wurde ein PC-basierendes System mit laufender VDR-Software geschaffen. Der Aufbau des Systems an der TU Ilmenau kann dem Anhang A entnommen werden. Detaillierte Informati-

[6] Vgl. [Stre 2004] S. 87 ff
[7] Vgl. [DVP 2005] Home

onen findet der interessierte Leser in der Dissertationsschrift[8] sowie in diversen Studien- und Diplomarbeiten, die im Laufe der Zeit im Rahmen des DVP-Projektes entstanden sind[9].

2.1.2 Das FORE-Modell

Das Konzept der Systemfamilien[10] wurde entwickelt, um den Aufwand von Einzellösungen zu reduzieren, indem Software wiederverwendet wird. Die Grundüberlegung ist, dass Softwareteile sich oft in Aufbau und Verwendung stark ähneln. Deswegen ist es sinnvoll und ökonomisch solche Potentiale zu nutzen. Zentrale Aufgabe ist die Wiederverwendbarkeit geeigneter Teile. Software soll nach diesem Prinzip aus einem festen Kern und variablen Teilen bestehen, um so Kundenanforderungen besser zu entsprechen. Um sie umsetzen zu können, müssen im Rahmen der Requirements Engineering Phase die bestehenden Anforderungen an die Software erhoben werden. Die erkannten Anforderungen werden in Modellen logisch angeordnet. Schließlich werden sie evaluiert, umgesetzt und entsprechend variiert oder verändert. Begleitend dazu muss der Aufwand geschätzt und Risiken bewertet werden. Der gesamte Prozess wird dabei dokumentiert. Zu diesem Thema werden verschiedene Konzepte, wie z.B. FAST, Feature Modellierung, FeatuRSEB und KobrA[11] in der Literatur beschrieben. Veränderbarkeit, Familienorientierung, Konfigurationsmöglichkeiten, das Spektrum der Parameter sowie die Überprüfbarkeit sind die Aufgaben, an denen sie sich differenzieren. Die Analyse dieser Konzepte erfolgte ausführlich in der Dissertationsschrift[12]. Problematisch an solchen Konzepten ist die Tatsache, dass sie nur mehr oder minder die jeweilige Aufgabe unterstützen. Um jedoch ein konsistentes Modell zu schaffen, müssen alle genannten Aufgaben logisch verknüpft und vollständig gelöst werden.

Die Beschreibung einer Systemfamilie erfolgt dafür in einem Merkmalmodell. Hier ergeben sich durch die Struktur der Systemfamilie jedoch Probleme. Denn für die Ableitung einer Systemvariante müssen sowohl Merkmale als auch Anforderungen und deren Beziehungen im Modell hinterlegt werden. Die Eindeutigkeit dieser Beziehungen ist dabei das Entscheidende. Ein in sich konsistentes, erweitertes Merkmalmodell ermöglicht so Varietät. Jedoch müssen auch im Vorfeld des Modells die erhöhten Anforderungen Beachtung finden. So müssen zusätzlich Dokumente und Personen im zeitlichen Verlauf festgehalten werden.

---

[8] Vgl. [Stre 2004] S. 19 ff

[9] Eine logisch zusammenhängende Übersicht der bisherigen Arbeiten kann [DVP 2005] Results oder [Stre 2004] S. 168 ff. entnommen werden.

[10] Eine ausführlichere Darstellung dieses Konzeptes erfolgt in Kapitel 3.2.1.

[11] Mehr Informationen unter [Stre 2004] S. 72 ff

[12] Vgl. [Stre 2004] S. 68 ff

Die Lösung der genannten Probleme erfolgt im FORE-Modell im Hinblick auf die Automatisierbarkeit und Überprüfbarkeit der Varianten-Ableitung. So werden im Modell Versionen im Rahmen der Systemfamilie eingeführt und Beziehungen zwischen den erhobenen Anforderungen dargestellt. Das FORE-Modell erweitert unter Nutzung der traditionellen Konzepte das Merkmalmodell. Die Daten werden durch das Anforderungsmodell und das erweiterte Merkmalmodell erhoben. Durch Nutzung des Entwicklungsprozesses fließen die Daten in das Datenmodell ein. Dieses schließt den Prozess des Requirements Engineering ab und es erfolgt der Entwurf der Software. Die einzelnen Schritte haben spezifische Aufgaben:[13]

Anforderungsmodell: Das Modell beinhaltet die hierarchisch angeordneten Anforderungen an das System und die entsprechenden Personen als **Personenmodell**, sowieso das referenzierte **Dokumentmodell,** welches die Struktur des Spezifikationsdokumentes[14] enthält. Die Beziehungen und Konflikte zwischen den Anforderungen müssen ebenso abgebildet werden. Sie können sich unter Umständen gegenseitig ausschließen. Um Änderungen nachvollziehen oder rückgängig machen zu können, bedarf es einer Aufzeichnung der Änderungen, der Historie. Um Anforderungen erheben zu können, wird eine Anforderungskarte benutzt. Sie protokolliert wichtige Informationen wie unter anderem Abhängigkeiten, den Autor, die Zuordnung zu einer Komponente oder ihre Bedeutung für den Kunden.

Merkmalmodell: Kernelemente in dem sich anschließenden Merkmalmodell sind die Beziehungen der Anforderungen zu den Merkmalen. Hier werden der Kern der Softwarefamilie mittels obligatorischer und die Varianten mittels optionaler Merkmale beschrieben. Merkmale sind als Zusammenfassung der Anforderungen an das System und somit als Eigenschaftsbeschreibungen zu verstehen. Sie können unterschiedliche Intentionen haben: Sie können funktionaler Natur sein, Schnittstellen ausdrücken, Parameter darstellen oder der Strukturierung des Modells dienen. Auch für ihre Erhebung wird eine Merkmalskarte genutzt, die ähnlich aufgebaut ist wie die Anforderungskarte, hier werden alle Informationen zu einem Merkmal kompakt zusammengefasst. Die Darstellung der Merkmale erfolgt hierarchisch, um ihre Beziehungen untereinander und zu der Systemarchitektur zu verdeutlichen. Sie können z.B. Verfeinerungen oder Einschränkungen anderer Merkmale aufzeigen. Zur Darstellung der zahlreichen Beziehungsmöglichkeiten wurde eine Notation geschaffen, sie wird in Anhang B gezeigt. So kann angezeigt werden, welche Varianten möglich sind und welche Merkmale sich ausschließen, sowie welche Merkmale zum Kern gehören, also notwendig sind, und welche für die Variante vom Kunden dazu gewählt werden können. Schließlich kann nur ein in sich

[13] Vgl. [Stre 2004] S. 87 ff

[14] Das Dokument enthält eine komplette Beschreibung der Systemfamilie einschließlich der Personen, der Anforderungen, der Varianten und deren Merkmale aus dem Merkmalmodell, sowie die Komponenten und Schnittstellen und dazu Kosten- und Risikobetrachtungen.

widerspruchsfreies Modell automatisch überführt werden. Diese Überprüfung erfolgt anhand automatischer Regeln.

Entwicklungsprozess: Der Entwicklungsprozess wird in der Literatur durch das „Six-Pack-Modell“[15] dargestellt und wird für die Generierung des Datenmodells genutzt. Dazu werden die Ergebnisse der **Domänenanalyse**, also dem Requirements Engineering der Softwarefamilie selbst und die Ergebnisse der **Applikationsanalyse**, also die Forderungen der Kunden an die Software, genutzt und abgeglichen. Das Resultat fließt in das **Anforderungsmodell** ein. Dieser Abgleich hat enorme Bedeutung, da ab hier Änderungen im Anforderungsmodell nur noch mit zusätzlichen Kosten und Aufwand zu bewältigen sind. Das FORE-Modell umfasst die genannten drei Module als ein Teil des Six-Pack-Modells, welches dem Systementwurf vorgelagert ist. Das Requirements Engineering der Systemfamilie und der kundenspezifischen Applikation ist ein weitreichender Prozess, hier wird auf die Dissertationsschrift verwiesen.

Datenmodell: Das Datenmodell ist die schriftliche Komprimierung der Resultate vergangener Schritte. Das Modell beinhaltet eine Sammlung der vollständigen Informationen. Das sind das **Anforderungsmodell**, das **Merkmalmodell**, das **Systemfamilienmodell**, das **Dokumentmodell**, das **Personenmodell** und ihre Beziehungen untereinander. Ziel ist es hier, die vielfältigen Beziehungen untereinander in einem UML-Diagramm darzustellen. Die Elemente selbst werden in einer Datenbank unter der entsprechenden Variante der Systemfamilie gespeichert. Das Anforderungsmodell stellt die Ansprüche an das System systematisch gegliedert dar, ihnen werden Personen, Dokumente und zahlreiche Informationen zugeordnet. Die Anforderungen werden dem Kern bzw. einer oder mehrerer Varianten zugeschrieben. Ähnliches geschieht mit den Merkmalen im Merkmalmodell. Hier liegt der Fokus vor allem in der Zuweisung der Art der Merkmale und der Frage, ob das Merkmal jeweils optional oder obligatorisch ist. Im Dokumentmodell schließlich sind alle erstellten Dokumente enthalten und das Personenmodell stellt alle Personen im Zusammenhang mit der Erhebung dar. Schließlich werden alle Änderungen mit dem entsprechenden Datum und der ausführenden Person in der Historie aufgeführt, die Erhebung der Merkmale ist damit abgeschlossen.[16]

[15] Das Six- Pack- Modell ist ein globales Modell der Systemfamilienentwicklung, vgl. [Stre 2004] S. 211. Das Modell einschließlich des gesamten Entwicklungsprozesses soll hier nur kurz erwähnt bleiben. Ausführlichere Erläuterungen sind [Stre 2004] S. 70 ff zu entnehmen.

[16] Vgl. [Stre 2004] S. 87 ff

2.2 Live-CD Projekte

Eine Live-CD ist eine bootfähige CD mit einem Betriebssystem, oft linuxbasierend, die zum Arbeiten weder eine Festplatte noch eine Partition benötigt. Sie läuft vollständig und direkt von CD, DVD oder einem USB-Stick; es ist keine Installation nötig. Das Betriebssystem schreibt sich lediglich in den Arbeitsspeicher und nutzt so eine virtuelle Festplatte. Konfigurationen können jedoch vorgenommen werden und werden dann auf Diskette oder USB-Stick gespeichert. Das Programm kann zudem auf Datenträger wie Festplatten, USB-Sticks, Disketten oder CDs zugreifen. Möglich macht das unter Linux die ausgereifte Hardwareerkennung Kudzu[17]. Zudem ist ihr Einsatz völlig unabhängig von bereits installierter Software auf dem PC. Eine aufwändige und mitunter zeitintensive Installation ist nicht erforderlich. Sie bieten normalerweise ein Gesamtpaket an Anwendungssoftware, die so unverbindlich mit dem Betriebssystem getestet werden können, oder auch eine bestimmte Aufgabe wie z.B. Virenscannen erfüllen. Sie können somit überall zum Einsatz kommen und werden so zum „individuellen Unterwegs-Büro“[18].

In dieser Diplomarbeit ist eine Live-CD wie folgt definiert:

Definition einer Live-CD[19]
Eine Live-CD ist ein bootbares Betriebssystem, oft mit Desktopumgebung, auf CD oder einem anderen bootbaren Medium, welches zum Arbeiten keine Festplatte oder ähnliches sowie kein anderes installiertes Betriebssystem benötigt, sondern temporär den RAM belegt und sich selbst verwaltet. Eine solche CD bietet eine Sammlung an Anwendungssoftware nach einem bestimmten Fokus. Sie ist keine reine Boot- oder Hilfsprogramm-CD.

Tabelle 1: Definition einer Live-CD

[17] Vgl. [Hatt 2005] S. 15
[18] Vgl. [Hatt 2005] S. 20
[19] Vgl. [Wiki 2005] Live-CD und [Hatt 2005] S. 13 ff

Die CDs bieten so eine große Zahl an Einsatzszenarien, die hier zusammengefasst sind:[20]

Einsatzmöglichkeiten von Live-CDs
• Anwendungsoberfläche • Gefahrlose Internetnutzung • Bildungs- und Schulungszwecke • Sicherheitsprogramme für Netzwerke • Unterhaltungszwecke: Spiele, Film, Musik • Fremd- und mehrsprachige Distributionen • Regionale Inhalte • Schwerpunkt-CDs mit Softwaresammlungen zu bestimmten Themen wie z.B. Medizin, Juristik, Biologie • Diagnose-CDs für Hardwaretests • Einsatz als Firewall und / oder Server • Wartungs-CDs • Rescue- und Recovery-CDs bei Virenbefall oder Systemausfällen • Werbezwecke • Gefahrlose Test- und Demonstrationsversionen für verschiedene Betriebssysteme und Anwendungssoftware • Komplettinstallationen von vollständig konfigurierten Linux-Systemen • Oder auch einfach zum Testen von anderen Betriebssystemen

Tabelle 2: Einsatzmöglichkeiten von Live-CDs

Da die Software selbst auf der CD nicht veränderbar ist, sind Live-CDs sicher in der Anwendung.[21] Allerdings benötigen die CDs oft mehr Geduld. Denn ein System, welches immer wieder erneut booten, Hardwareerkennung und Konfiguration durchführen muss, braucht für den Start und die Ausführung der Programme länger und erreicht einen konstanten Geräuschpegel. Zudem verfallen mit jeder Sitzung alle Systemeinstellungen und Daten, wenn nicht selbst an die Rücksicherung per USB-Stick, Festplatte oder CD gedacht wird.

Auch ist bei Linux Live-CDs ein schreibender Zugriff auf Windows-Partitionen mit NTFS nicht möglich. [22]

[20] Vgl. für diese Tabelle [Live 2005] Startseite. Diese Liste erhebt nicht den Anspruch, vollständig zu sein.
[21] Vgl. [Wiki2005] Live-CD
[22] Vgl. [Kofl 2003] S. 1258

Eine Zusammenfassung der Vor- und Nachteile von Live-CDs sei hier dargestellt[23]:

Bewertung einer Live-CD[24]	
Vorteile	**Nachteile**
• Gefahrloser, schneller Einblick in Linux • Laufendes Betriebssystem ohne Installationshürden • Festplatten etc. spielen keine Rolle • Ausgeklügelte Komprimierungsstrategien ermöglichen eine enorme Zahl möglicher Programme • Zahlreiche Einsatzmöglichkeiten, gerade im Bereich Trouble-Shooting (Viren, Sicherheit, Abstürze etc.) • Kostenlose Verfügbarkeit • Aufgrund der Vielzahl an Distributionen kann für jedes System, auch sehr alte Rechner, ein passendes Betriebssystem gewählt werden • Live-CDs können den eigenen Ansprüchen angepasst und auch an die Community weitergegeben werden • Die CD ist portabel und kann flexibel eingesetzt werden • Die automatische Hardwarekennung ermöglicht die Nutzung von Komponenten ohne zusätzliche Treiberinstallation • Bereits installierte Betriebssysteme und Daten werden nicht berührt • Alle Medien, die BIOS als bootbar interpretiert, können für ein Live-System verwendet werden • Gestattet eine einfache Installation • Risikolose Internetnutzung ohne Virengefahr	• Eingewöhnungszeit • Die Dauer des Systemstarts erfordert Geduld • Das Arbeiten gestaltet sich relativ „zäh“ • Das Speichern von Daten und Einstellungen und deren Einspielung muss bei einem erneuten Start eingeplant werden • Leise Laufwerke sind von Vorteil, da die Arbeit mit dem CD-Laufwerk oftmals einen konstanten Geräuschpegel erreicht • Schwierige Installation neuer Programme für unerfahrene Nutzer

Tabelle 3: Bewertung von Live-CDs

Live-CDs gibt es für viele Betriebssysteme; auch für Exoten wie BeOS, ReactOS oder AFROS werden solche CDs entwickelt. Eine kurze Übersicht mit weiterführenden Links gibt es unter [Wiki 2005] Live-CD.[25]

In den folgenden Unterkapiteln seien Live-CD Projekte der beiden am weitesten in Deutschland verbreiteten und für diese Diplomarbeit relevanten Betriebssysteme vorgestellt.

[23] Alle genannten Punkte konnten im Rahmen dieser Diplomarbeit bestätigt werden.

[24] Vgl. [Hatt 2005] S. 13 ff

[25] Mehr Informationen zum Thema gibt es unter [Live 2005] Startseite, den Foren und dem entsprechenden Wiki.

2.2.1 Bestehende Live-CD Projekte unter Linux

Linux Live-CDs, auch humorvoll „Little Live Linuxes“[26] genannt, erfreuen sich großer Beliebtheit. Deswegen gibt es momentan rund 250 verschiedene Distributionen. Umfangreiche Listen an Beschreibungen und Downloads der Live-CDs auf der Grundlage von Linux gibt es unter [Live 2005] Startseite oder [Bodn 2005].

Solche Distributionen eignen sich zum Remastern, dem Modifizieren der CD nach eigenen Anforderungen. Im Internet gibt es dafür diverse Anleitungen. Die Erweiterbarkeit von Linux Live-CDs ist über 3 verschiedene Wege möglich:

1. Erweiterung durch Softwarepakete („Minimalansatz“)
2. umfangreiches Modifzieren einer CD durch De- und Installation von Softwarepaketen („Maximalansatz“)
3. Eigenständige Erstellung einer Live-CD (z.B. mit Hilfe von Toolkits)

In der Studienarbeit von Melanie Schöppe und Martin Rulsch werden diese 3 Wege ausführlich beschrieben.[27] Zwei bekannte Distributionen, die für diese Arbeit relevant sind, seien im Folgenden vorgestellt.

2.2.1.1 Knoppix 3.9[28]

Knoppix, benannt nach dessen Entwickler Klaus Knopper, ist eine freie Linuxdistribution auf der Basis von Debian GNU / Linux[29]. Die Software startet direkt von CD oder DVD und erkennt dank einer gut gepflegten Treiber-Datenbank nahezu jede von Linux unterstützte Hardware schon beim Start automatisch. Die Hardware kann ebenso konfiguriert werden. Falls einmal exotische Hardware benutzt wird, gibt es universelle Treiber, die vom Nutzer angepasst werden können. Das Software-Paket auf der Live-CD bringt bereits eine große Anzahl an Programmen mit, die für normale Ansprüche vollkommen ausreichend sind. Es enthält unter anderem die grafische Benutzeroberfläche KDE mit vielen Zusatzprogrammen für den täglichen Gebrauch wie OpenOffice.org, Gimp, Mozilla, Xchat und Gaim. Erfahrene Linux-Nutzer können das Knoppix-Betriebssystem darüber hinaus auch direkt auf der Festplatte installieren. Diese Eigenschaft macht Knoppix zu einer der am schnellsten einrichtbaren Distributionen.[30] Da Knoppix für alle frei zugänglich ist und dank des frei kopierbaren Debian-

[26] Vgl. [Lin-G 2005] Node 9185

[27] Vgl. [Schö 2005] S. 1 ff

[28] Vgl. [Knop 2005] Knoppix, [Wiki 2005] Knoppix und [Kofl 2003] S. 1257 ff

[29] Vgl. [Wiki 2005] Debian: Debian ist eine GNU / Linux-Distribution aus ausschließlich freier Software. Debian enthält das Betriebssystem und eine große Auswahl an Anwendungsprogrammen, Tools und Utilities, zusammen mit einem passenden Kernel.

[30] Vgl. [Lin-T 2005] Knoppix FAQ

Systems lizenzfrei verwendet werden kann, wird die Software oder die automatische Hardwareerkennung Kudzu gerne für andere Derivate verwendet. Entsprechend der Anforderungen werden dann für Nutzerzielgruppen Programme installiert. Besonders vorteilhaft dabei ist es, dass dank transparenter Kompression bis zu 2 GB auf der CD installiert werden können. Einige bekannte Derivate von Knoppix seien im Nachfolgenden vorgestellt:

Knoppix -Derivate	
Morphix	Ein modular aufgebauter Knoppix-Abkömmling.
Kanotix	Basiert auf Knoppix und fokussiert eine optimale Installation.
Gnoppix	Ein Knoppix mit GNOME-Oberfläche.
Knoppix For Kids	Eine spezielle Variante für Eltern und Kinder.
Knoppix STD	Bündel an Sicherheitschecks wie z.B. die Schwachstellensuche
Knoppicillin	Eine menügesteuerte Sammlung von Virenscannern.
Gibbix	Ein Knoppix-Derivat für Schulungszwecke.
Skolelinux	Ein Knoppix-Derivat für Schulungszwecke.
EDU Knoppix	Ein Knoppix-Derivat für Schulungszwecke.
Games-Knoppix	Ein Knoppix für Spiele mit erweiterter Hardwarebeschleunigung.

Tabelle 4: Knoppix Derivate

2.2.1.2 Kanotix 2005-03[31]

Dieses Knoppix-Derivat legt den Fokus auf besonders gute und aktuelle Hardwareerkennung, eine elegante und unkomplizierte Installation auf eine Festplatte und möchte die Unzulänglichkeiten von Knoppix, wie z.B. die unzureichende Unterstützung von WLAN-Karten, beseitigen[32]. Kanotix bringt ebenfalls ein großes Paket an Anwendungssoftware mit. Um die Aktualität zu sichern, werden bei Kanotix auch gerne Pakete verwendet, die sich noch im Testmodus befinden oder noch als instabil gelten.

[31] Vgl. [Kano 2005] Info, [Kan-W 2005] und [Wiki 2005] Kanotix

[32] Vgl. [Heis 2005] Newsticker 60178; Im Rahmen dieser Diplomarbeit konnte die Meinung vieler Kanotix- und Knoppix- Nutzer in diversen Foren bestätigt werden: Kanotix ist tatsächlich in der Hardwareerkennung unschlagbar.

2.2.2 Live-CD Projekte unter Windows

Auch unter Windows gibt es eine offizielle Live-CD, die jedoch nicht frei zugänglich ist. Deswegen wurde diese CD für Privatnutzer modifiziert. Um rechtliche Probleme zu umgehen, wurde dafür ein Programm geschrieben, welches Besitzern einer Original-CD eines Windows-Betriebssystems die Erstellung einer eigenen Live-CD ermöglicht.

2.2.2.1 Microsoft Windows PE

Windows PE ist eine Live-CD, bei der es sich um eine 130 MB große bootende Windows Variante von XP oder dem Windows 2003 Server handelt. Die CD kann jedoch nur von Microsoft Großhändlern bezogen und genutzt werden. Dort wird die CD zur Erstinstallation von Windows und Anwendungsprogrammen bei neuen PCs für den Handel verwendet.

2.2.2.2 Bart PE[33]

Bart PE, bzw. Bart's PE, ist die „Für-Jedermann-Version" von Windows PE, die es seit April 2003 gibt. Dazu gehört eine 150 MB große Live-CD mit einem Windows-Reservesystem und flexiblen Plugins sowie der Möglichkeit, zusätzliche Tools mit auf die CD zu brennen. Zur Erstellung dieser CD ist das Programm PE Builder von Bart Lagerweij erforderlich. Die Bart PE Software selbst stammt also nicht von Microsoft, muss aber, da legal, gebilligt werden, denn jeder, der über ein laufendes Windows 2000 und höher und die dazugehörige Installations-CD verfügt, kann mit der Software eine eigene Live-CD mitsamt Netzwerkunterstützung und Remotezugriff sowie grafischer Oberfläche erstellen. Das System ist dabei offen konzipiert und lässt sich individuell anpassen; andere Programme können in das zu brennende Verzeichnis mit hinein kopiert werden, eine zusätzliche XML- und eine Info-Datei sorgen für die nötigen Verknüpfungen und Systemeinstellungen für die Registry. Zudem können im Internet diverse Plugins bezogen werden. Besonderer Wert wird hierbei darauf gelegt, dass Bart PE Daten von NTFS und Fat32 Partitionen lesen und schreiben kann. Mit der CD können Windowsinstallationen repariert, kopiert oder bearbeitet und Daten gesichert werden. Die Standardumgebung wird als Mini-Windows transportabel, gerade für Administratoren ein nützliches Programm. Bart PE bietet somit alle Vorteile von Linux Live-CDs.

[33] Vgl. [Lage 2004]

2.3 Digitalfernsehen

Seit den ersten Grundüberlegungen Anfang der 1990er Jahre und dem Beginn der Umstellung von analog auf digital durch die in Deutschland 1993 gegründete „Gesellschaft zur Förderung der Rundfunkversorgung“, kurz GARV,[34] erobert digitales Fernsehen die deutschen Haushalte. Auch europaweite Bestrebungen gründen sich im Jahr 1993, als das DVB-Projekt in der Schweiz gestartet wurde[35]. Bis 2010 sollen nun alle deutschen Haushalte Digitalfernsehen über einen einheitlichen Standard empfangen können. Der digitale Standard verbessert das Fernsehen in vielerlei Hinsicht durch erhöhte Übertragungskapazitäten[36] und eine Vielzahl an Zusatzinformationen und Diensten.[37] Die Vorteile seien hier zusammenfassend dargestellt:

Vorteile des Digital TV[38]
Bessere Übertragung: Die Digital-Technik ermöglicht durch Komprimierungsalgorithmen eine höhere Informationsübertragung: Bessere Bilder, besserer Ton, mehr Kanäle bei weniger Störanfälligkeit. Die besten Voraussetzungen für das Heimkino-Feeling, denn bei einigen Filmhighlights wird der Ton im Dolby-Digital-Format ausgestrahlt.[39]
Mehr Leistungen: Durch den digitalen Übertragungsweg sind auch neue Verschlüsselungsstandards entstanden. Sie ermöglichen eine Vielzahl von Pay-TV Angeboten wie z.B. Video-On-Demand, oder eine schier unglaubliche Zahl an Sendern, auch ausländische in DVD-Qualität mit vielen interaktiven Zusatzinformationen und dazu internationale Newsticker[40], Emailfunktionen und elektronischen Einkauf, individuelle Video-Informationen über andere Länder oder Urlaubsziele, Live Chats und Spiele.
Mobiler Empfang: Die Übertragung per Antenne bietet das „Überall-Fernsehen“ kabellos. Egal ob Laptop, Auto oder Zeltplatz, PDA, Handy oder Smartphone, digitales Fernsehen kann überall genutzt werden.

Tabelle 5: Vorteile des digitalen Fernsehens

Das digitale Signal kann über folgende Wege empfangen werden: Per Satellit (DVB-S), per Kabel (DVB-C) oder per Antenne (DVB-T). In den meisten Ballungsgebieten ist inzwischen

[34] Die Gesellschaft wurde am 22.Dezember 1993 gegründet, mehr dazu unter [GARV 2005] Home.

[35] Unter der Homepage des Projektes gibt es viele, auch weltweite Informationen über den DVB- Standard. Vgl. [DVB-O 2005] Home

[36] Komprimierungsmöglichkeiten wie der MPEG2- Standard lassen bis zu 96% der Datenmenge einsparen. Damit können auf einer Frequenz inzwischen bis zu 10 digitale statt einem analogen Programm gesendet werden. Genauso werden Frequenzen bei den Tönen abgeschnitten, um die Datenmenge zu reduzieren.

[37] Vgl. [Dig-F 2005] Einleitung

[38] Vgl. hierzu [Ue-TV 2005] Das ÜberallFernsehen und [CT-SO 2004] S. 118 ff

[39] Vgl. [CT-SO 2004] S. 119

[40] Anbieter sind z.B. [Kab-D 2005] oder [Kab-B 2005]

der Empfang kein Problem mehr. Zusätzlich können umfangreiche Serviceangebote oder eine Vielzahl an Sendern bezogen werden.

Digitalfernsehen wird von wirtschaftlicher Seite stark unterstützt. Das „Digital Video Broadcasting Project“[41], in dem über 270 Unternehmen wie Rundfunkbetreiber, Hersteller, Kabel- und Softwarefirmen und Behörden aus 35 Ländern zusammenarbeiten, schafft seit vielen Jahren einheitliche Standards. Digitales Fernsehen ist europaweit, genauso wie in Asien und Südamerika seit langem ein aktuelles und profitables Thema.

Digitalfernsehen kann auf verschiedenen Wegen empfangen werden[42]:

	Analoge Endgeräte	**Digitale Endgeräte**	**PC**
Digitaler Empfang	Über Set-Top-Box zur Umrechnung des digitalen Signals	Über direkte Verbindung	Über eine DVB-Karte oder per USB von einer Set-Top-Box und entsprechende Software
Digitale Aufnahme auf CD oder DVD	DVD-Recorder Festplattenrecorder Festplattenreceiver	DVD-Recorder Festplattenrecorder Am PC per DVI-Anschluss	Per Software auf Festplatte
	Kapitel 2.3.1		Kapitel 2.3.2

Tabelle 6: Empfang des Digitalfernsehens

2.3.1 Digitalfernsehen im Wohnzimmer

Wer digitales Fernsehen am analogen Endgerät nutzen möchte, braucht zusätzliche Komponenten, die das digitale Signal verarbeiten und entschlüsseln können[43]. Für die Übersetzung wird eine Set-Top-Box benötigt[44]. Moderne Integrated Digital TVs, kurz IDTV[45], haben so eine Box bereits eingebaut. Wer Aufnahmen tätigen möchte, benötigt einen DVD-Recorder, der dann die analogen Bilder und Töne auf CD oder DVD abspeichert. Eine andere Möglichkeit sind Festplattenrecorder bzw. Festplattenreceiver, sie sind Aufnahmegerät und Set-Top-Box in einem. Sie speichern die Daten auf einer internen Festplatte[46]. Recorder, die

[41] Vgl. [DVB-O 2005] Home
[42] Vgl. [CT-SO 2004] S. 28 ff
[43] Vgl. [Dig-F 2005] Einleitung
[44] Vgl. [DVBT 2005] FAQ; Set-Top-Boxen werden oft auch DVB- oder Digital- Receiver genannt.
[45] Vgl. [Ue-TV 2005] Geräte
[46] Eine Übersicht über derzeitig angebotene Festplattenrecorder kann unter [Fest 2005] nachgelesen werden

DVDRAM unterstützen, bieten die Time-Shift-Funktion[47]. Die propritären Aufnahmeformate machen aber oft ein Weiterverwenden der Sendungen z.B. am PC unmöglich[48]. Oft stößt man dabei auch schnell an die Grenzen der internen Festplatten.

2.3.2 Digitalfernsehen am PC

Für das Digitalfernsehen ist auch der PC, sofern mit den entsprechenden Leistungsmerkmalen vorhanden, als Multimediaanlage zu Hause geeignet. Ein solcher HTPC ist in der Lage verschiedenste Multimediadaten wie z.B. MP3s, VCDs, DVDs, DivX und AVI Dateien abzuspielen. Das Gerät fasst damit Einzelgeräte wie z.B. Videorecorder, DVD-Spieler, CD-Spieler, Hifi-Anlage in einem Gerät zusammen und ermöglicht dabei noch weitere Anwendungsmöglichkeiten.[49] Alles was hierfür benötigt wird ist eine DVB-Karte oder eine per USB angeschlossene Set-Top-Box mit Digital-Ausgang und einem Anschluss per Kabel, Satellit oder Antenne. Leider sind manche Karten im Bereich der DVB-T noch etwas unausgereift; für viele Karten sind wahre „Patchorgien“ notwendig, um die Geräte lauffähig zu machen. Dazu gehört die entsprechende Software, die oft den Karten beigefügt, aber manchmal noch sehr unkomfortabel ist[50]. Deswegen sind in letzter Zeit verschiedene Opensource-Projekte entstanden, bei denen mit Hilfe der engagierten Entwicklergemeinde kostenlose und komfortable Software für alle Ansprüche erstellt werden soll. So wird der PC zur Alternative gegenüber dem klassischen Fernsehgerät.

Ausgewählte bekannte und damit stark forcierte Opensource-Projekte für Linux seien hier im Folgenden vorgestellt.

[47] Zeitversetztes Fernsehen ist eine beliebte Funktion bei digitalen Videorecordern, bei der eine Sendung aufgenommen wird, während nur wenige Minute später bereits mit dem Anschauen begonnen werden kann und die Aufnahme im Hintergrund weiter läuft.

[48] Vgl. [CT-SO 2004] S. 28 ff Die Weiterverarbeitung der Daten ist aufgrund inkompatibler Datentypen nicht ohne weiteres möglich. Der Hersteller Seagate nennt dieses Verfahren „DriveTrust-Technik“. Vgl. hierzu [PC-MA 2005] Nachricht 37280

[49] Vgl. [Wiki 2005] HTPC

[50] Vgl. [CT-SO 2004] S. 125 ff, Leider ist das auch 2005 noch so, wie im Rahmen dieser Diplomarbeit festgestellt werden musste.

2.4 Software-Projekte für digitales Fernsehen

2.4.1 Linuxbasierende Projekte

2.4.1.1 Das VDR-Projekt

Die VDR-Software wurde 2002 von Klaus Schmidinger ins Leben gerufen[51] und kann kostenlos in der Version 1.2.6 von der Homepage des VDR-Projektes[52] heruntergeladen werden[53.] Die Software ermöglicht es, Fernsehen und Radio in hoher Bild- und Tonqualität unter Linux aufzuzeichnen. Der Leitgedanke war es, einen linuxbasierenden, freien, digitalen Videorecorder zu entwickeln der pluginbasiert aufgebaut ist, und anders als die herkömmlichen kommerziellen Produkte allen Ansprüchen gerecht werden soll, indem flexibel jeder Nutzer seine für ihn notwendigen Plugins in das Grundsystem des Recorders einbindet. Der Source-Code der Software ist freigegeben, so dass es inzwischen eine Vielzahl Plugins gibt, die von der interessierten Entwicklergemeinde programmiert wurden. Bisher wurden 76 Plugins auf der Homepage des VDR-Portals der Öffentlichkeit vorgestellt. Diese und weitere verfügbare Plugins aus anderen Sammlungen sind zur Information im Anhang C aufgelistet. All diese Erweiterungen können in die VDR-Software eingebunden werden und machen damit das Leistungsspektrum von VDR umfangreicher als in manchen Fertiggeräten[54]. Im Oktober 2004 wurde die kostenlose Software sogar zum Testsieger im PC Magazin unter fünf Media Center Programmen gekürt.[55]

Was die Software alles kann, sei hier beispielhaft dargestellt[56]:

[51] Vgl. [VDR-P 2005] VDR Info

[52] Vgl. [Schm 2005] Download

[53] Die aktuelle Testversion für Entwickler ist die 1.3.24, vgl. [VDR-P 2005] Download

[54] Inzwischen findet man im Internet komplette Anleitungen, sich eine eigene VDR- Box fürs Wohnzimmer mit LCD, Gehäuse mit Netzteil, Festplatte, Fernbedienung, Lärmschutz, Verstärker, Motherboard etc. selbst zu bauen. Solche Boxen werden auch unter den Namen Reelbox (Vaporware), Topfield oder Dreambox kommerziell vertrieben. Vgl. [VDR-P 2005] VDR Info und [Heim 2005] Home

[55] Vgl. [PC-MA 2005] Softwaretests

[56] Vgl. für diese Tabelle [VDR-P 2005] VDR Info

Funktionalitäten von VDR
• Abspielen von AVI, MPEG, DVDs, VCDs, SVCDs über den TVout der DVB-T Karte. • Abspielen von Audio CDs mit CDDB Support • Umwandlung der Aufnahmen in ein platzsparendes Format per OSD • Timeshift mit einer vollwertigen Karte • Parallele Aufnahme mehrerer Sendungen auf gleicher Frequenz oder bei mehreren Karten • Komfortables Programmieren des VDRs per Internet oder LAN • MP3 Support per OSD • DVD Navigationsmenü • Videotext Support • Konvertieren von Mitschnitten, direkt nach der Aufnahme per OSD Menü in ein komprimiertes Format wie VCD, SVCD, MPEG oder DIVx • Schneiden der Aufnahmen direkt per Fernbedienung um z.B. Werbepausen zu entfernen • Streaming des Signals via LAN • Optische Ausgabe über ein Display • Wiedergabe von Bildern • Definieren von Befehlen (Ausführung per OSD)

Tabelle 7: Das Leistungsspektrum der VDR-Software

Da die Software so beliebt ist, wurden inzwischen auch einige Distributionen basierend auf Linux zusammengestellt und seien hier kurz aufgelistet[57]:

VDR-Distributionen	
Installation	**Live-CD**
LinVDR C'T-VDR Mulimidix Gen2VDR VDR inside VDR4You MiniVDR MiniDVBLinux	GeexBox VDR inside VDRlive VDRLiveCD

Tabelle 8: VDR Distributionen

[57] Diese Tabelle wurde dem [VDR-W 2005] VDR-Distributionen entnommen.

Diese Distributionen unterscheiden sich in drei wichtigen Kriterien. Sie enthalten verschiedene Zusammenstellungen von Plugins bzw. unterstützen bestimmte Hardware. Dadurch besitzen sie auch unterschiedliche Größen von 32 MB für z.B. USB-Sticks[58], bis hin zum Inhalt einer kompletten CD. Die meisten Distributionen sind für eine Installation vorgesehen; sie enthalten eine „abgespeckte“ Linux-Distribution mit integrierter VDR-Software für eine schnelle Installation auf selbst zusammengestellten Hardwarekonfigurationen für Wohnzimmer oder PC. Die rechts in Tabelle 8 aufgeführten Distributionen sind Bemühungen VDR über eine echte Live-CD zugänglich zu machen. Zwei Bekannte Vertreter seien im Folgenden näher beschrieben.

a. LinVDR

LinVDR ist eine solche Live-CD für eine Installation mit integrierter VDR-Software und rund 40 Plugins und Zusatzprogrammen für eine komfortable Bedienung. Das zu Grunde liegende Betriebssystem ist stark reduziert; LinVDR ist nur rund 50 MB groß. Ziel war es, ein stabiles und trotzdem kleines VDR-Linux für eine einfache und komplette Installation inklusive automatischer Installationsroutinen zu entwerfen. Die aktuelle Version 0.7 kann unter [Koch 2004] heruntergeladen werden. Die Distribution unterstützt jedoch nur wenige DVB-Karten und ältere Versionen von Plugins.

b. C’T VDR

Auch C'T VDR, von der gleichnamigen Zeitschrift des Heise Verlages entwickelt, liefert ein stark reduziertes debianbasiertes Linux inklusive der VDR-Software und diversen Plugins und Installationsroutinen für eine problemlose und schnelle Installation. Die auftretenden Probleme sind dieselben wie bei LinVDR: Das letzte Update der Software erfolgte im Sommer 2004; Kernel, Plugins und somit auch die unterstützten Karten sind nicht mehr aktuell. Die Distribution ist aber für alle ideal, die sich eine eigene VDR-Box zusammenbauen möchten. Die aktuelle Version 3 kann unter [Hei-F 2004] bezogen werden.

[58] Unter [Mueh 2005] gibt es ein How-To für LinVDR auf einem USB-Stick.

2.4.2 Windowsbasierende Projekte

Aus der Vielzahl kommerzieller wie auch kostenloser Software für HTPCs seien hier stellvertretend vier in Deutschland bekannte Vertreter unter Windows genannt. Informationen sind unter den rechts genannten Links nachzulesen und sollen hier keine weitere Erläuterung finden, da das praktische Beispiel diese Diplomarbeit auf Linux basiert.

Software für Digitalfernsehen unter Windows	
Name	**Informationen unter**
Windows Media Center 2005	http://www.microsoft.com/windowsxp/mediacenter/evaluation/hdtv/default.mspx
GB-PVR	http://www.gbpvr.com/
SageTV	http://www.sage.tv
TVOON Media Center	http://www.tvoon.org/index.html

Tabelle 9: Software für Digitalfernsehen unter Windows

3 Konfigurationsmöglichkeiten von Software

3.1 Allgemeine Einführung

In diesem Kapitel sollen nun ausführlich Konfigurationskonzepte von Softwaresystemfamilien und von Software unter den beiden gebräuchlichsten Betriebssystemen vorgestellt werden. Ziel ist es, die Konfigurierbarkeit von Softwaresystemfamilien an gegebene Anforderungen zu prüfen und so eine optimale Vorgehensweise für die Umsetzung der VDR-basierten Software in Kapitel 4 zu erstellen. Deswegen wird auf verschiedene Grundideen für Software-Konfiguration eingegangen und ihre Vor- und Nachteile erläutert. Die Bewertung soll dazu dienen, die Nutzbarkeit und Anpassbarkeit dieser Grundideen aufzudecken und die Konfiguration der Live-CD weiter zu entwickeln.

Dabei wird im Rahmen der Arbeit folgendes unter dem Begriff „Konfiguration“ verstanden:

Definition: Konfiguration[59]
„Konfiguration“ beinhaltet die Einrichtung, Anpassung und Zusammenstellung interner und externer Komponenten eines Computersystems (Hardware-Sicht) oder die Zusammenstellung und Anpassung eines Programms auch mit Hilfe von Komponenten (Software-Sicht) an ein zukünftiges oder vorliegendes System oder eines Benutzers an die Gegebenheiten eines Systems für die Einstellung der Parameter des gewünschten Anwendungsfalls. Dies kann in allen Phasen der Softwareentwicklung geschehen.

Tabelle 10: Definition von Konfiguration

Definition Konfigurationsmanagement[60]
DIN EN ISO 10007 1996 - „Qualitätssicherung, Leitfaden für Konfigurationsmanagement“: Konfigurationsmanagement sind technische und organisatorische Maßnahmen zur Konfigurationsidentifizierung, Konfigurationsüberwachung, Konfigurationsbuchführung und Konfigurationsauditierung.

Tabelle 11: Definition von Konfigurationsmanagement

[59] Diese Definition orientiert sich an diversen Lexika: [INPR 2005] Fachwörter, [Hand 2005] Fachwörterbuch, [Unit 2005] Glossar, [Seib 2005] Glossar, [zpe-g 2004] und [Inte 2005] Glossar

[60] Zitiert nach [Qual 2005] QM- Lexikon und [DIN 10007]

Es gibt mehrere Konzepte Software konfigurierbar zu gestalten, die sich in den Softwareentwicklungsprozess eingliedern. Zur Einordnung in die entsprechende Softwareentwicklungsphase ist der Bindungszeitpunkt wichtig, in dem die variablen Anteile festgelegt werden. Dabei kann Konfiguration zu folgenden Zeitpunkten stattfinden:

1. Zur Entwicklungszeit über die Art und Weise des Codeaufbaus
2. Zur Compilerzeit über Preprozessor Anweisungen
3. In einer Konfigurationsdatei beim Einrichten und Starten eines Programms
4. Zur Laufzeit über die Einstellbarkeit

Um die Einordnung der verschiedenen Konfigurationskonzepte im Zeitverlauf des Softwareentwicklungsprozesses zu verstehen, sei dieser im Folgenden nach [Balz 2001] S. 51 ff dargestellt:

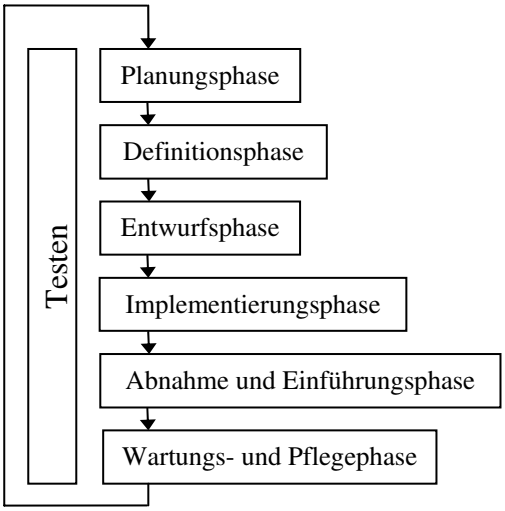


Abbildung 1: Phasen des Softwareentwicklungsprozesses

Zur Softwareentwicklung gehören die Aufgaben der Produktplanung, -definition, -entwurf und der Produktrealisierung. Dazu müssen die Qualitäts- und Kundenanforderungen erfüllt sein. Nach Erstellung der Software muss diese während der Anwendung gepflegt und gewartet werden. Dies beinhaltet z.B. Fehler zu beseitigen, die Software an neue Bedingungen anzupassen und neue Anforderungen an die Software zu erfüllen.[61]

[61] Vgl. [Balz 2001] S. 39

3.1.1 Ablauf des Softwareentwicklungsprozesses

Der Softwareentwicklungsprozess umfasst nach [Balz 2001] 6 Phasen, die in Abbildung 1 dargestellt sind.

Die Softwareentwicklung beginnt mit der **Planungsphase**, in der eine Untersuchung vorgenommen wird, ob sich das Softwareprodukt nach fachlichen, ökonomischen und personellen Gesichtspunkten realisieren lässt. Es werden hier diverse Studien durchgeführt, um das passende Produkt auszuwählen und schließlich grob Anforderungen, Funktionen, Qualitätsmerkmale etc. zu erheben. Schließlich muss in dieser Phase überprüft werden, ob sich das Projekt umsetzen lässt. Am Ende stehen das Lastenheft, ein Projektplan und eine Aufwandsschätzung[62].

Hier schließt sich die **Definitionsphase** an. In dieser Phase werden die anfangs noch sehr vagen Produktanforderungen spezifiziert, indem sie mit Hilfe verschiedener beteiligter Akteure ermittelt, deren Daten und Funktionen modelliert, analysiert und nach Simulation und Ausführung festgelegt werden. Ziel ist es, ein konsistentes und vollständiges Produktmodell zu entwickeln. Das Ergebnis dieser Phase ist ein Pflichtenheft, eine konkrete Zusammenfassung aller zu realisierenden Anforderungen, Funktionen, Daten und Strukturen, und ein Produktmodell oder ein oberflächlicher Prototyp.[63]

In diesen beiden Phasen ist der Begriff des Requirements Engineering verankert[64].

In der **Entwurfsphase** werden auf Grundlage der erfassten Anforderungen die Softwarearchitektur, die Nutzeroberfläche, Schnittstellen, Datenbankanbindungen und andere Zielplattformen entwickelt. Hier werden Rand- und Umfeldbedingungen berücksichtigt und die grundlegenden Entscheidungen z.B. über Schnittstellen zu peripheren Programmen getroffen. Die zu entwickelnde Softwarearchitektur beschreibt die Struktur des Software-Systems durch die Systemkomponenten und deren Beziehungen. Sie soll funktionale und nichtfunktionale Produktanforderungen und Qualitätsanforderungen erfüllen und Schnittstellen versorgen.[65]

Die Ergebnisse dieses Abschnitts sind die Vorraussetzungen der **Implementierungsphase**. Hier wird die entworfene Softwarearchitektur in Programmiertätigkeiten überführt, um die

[62] Vgl. [Balz 2001] S. 58 ff

[63] Vgl. [Balz 2001] S. 98 ff

[64] Vgl. [Stre 2004] S. 12 ff; Das Requirements Engineering wird nach [Balz 2001] S. 118 als ein Teilgebiet der Softwaretechnik verstanden, in dem Anforderungen ermittelt, beschrieben, modelliert und analysiert werden und wird im Rahmen dieser Diplomarbeit anhand der Aufgaben des Requirements Engineering beiden Phasen zugeordnet.

[65] Vgl. [Balz 2001] S. 719 ff

geforderten Leistungen der vorgegebenen Spezifikation zu implementieren. Dazu gehören die programmiertechnische Entwicklung der Software sowie die Dokumentation und Testfälle.

In der nun folgenden **Abnahme- und Einführungsphase** wird die fertiggestellte Software an den Auftraggeber oder Markt nach zahlreichen Tests und mit einem Abgabeprotokoll übergeben. Dann folgt bei Auftragsarbeiten die Einführung bei dem Kunden durch Installation, Schulung, Übergabe der Benutzerdokumentation und Inbetriebnahme der Software.[66]

Die **Wartungs- und Pflegephase** beginnt nach der Softwareeinführung. Herausforderungen sind die Fehlerkorrekturen, das Altern der Software und somit nötige Leistungsverbesserungen und das Anpassen von Änderungen oder Erweiterungen. Hier können auch bereits Informationen und Anforderungen an ein neueres Produkt gesammelt werden.[67]

Testen und Validieren sind integrierte Aufgaben, die sich durch alle Phasen ziehen.[68]

3.1.2 Konzepte der Systemfamilienentwicklung

Zur Bewerkstelligung der besonderen Anforderungen von Systemfamilien und der Wiederverwendung auf der Ebene des Codes, des Design und der Architektur von Software werden im Folgenden dazu Konzepte und Methoden[69] vorgestellt, die einander bedürfen und aufeinander im Laufe der Softwareentwicklung aufbauen.

Ausgehend von der Objektorientierung und deren **feingranularen Klassenstruktur** wird Code wiederverwendet. Spezialisierte Klassen über definierte öffentliche Schnittstellen lassen sich in **Klassenbibliotheken** zusammenfassen und wiederverwenden. Über **Vererbung** können neue Klassen abgeleitet werden, an die dynamisch Anfragen gestellt werden können. Der konkrete Einsatz zur Laufzeit von Klassen kann über den **Polymorphismus** oder **ifdef-Anweisungen**[70] gesteuert werden. Die Erstellung universeller Klassen bedarf dabei sorgfältiger Analysen und Merkmalmodelle, die jedoch das Prinzip der **Datenkapselung** verletzen, wenn periphere Funktionen wie Persistenz, Sicherheit, Remotezugriffe etc. in verschiedenen Klassen geändert werden. Dafür lässt sich die **Aspektorientierte Programmierung** einsetzen, wo solche kritischen Prozesse in eigenen Klassen zusammengefasst werden, die Basis-

[66] Vgl. [Balz 2001] S. 1085 ff

[67] Vgl. [Balz 2001] S. 1090 ff

[68] Das Testen beginnt nach [Balz 2001] S. 1065 in dieser Phase als eine der Aufgaben, in [Stre 2004] S. 14 f wird Testen als integrierte Phase verstanden, die sich kontinuierlich durch alle anderen zieht.

[69] Zur Definition einer Methode vgl. [Balz 2001] S. 36: Eine Methode ist eine planmäßig angewandte und begründete Vorgehensweise zur Erreichung von definierten Zielen im Rahmen festgelegter Prinzipien. Eine Methode wird dabei als Obergriff für Konzepte, Notation und methodischer Vorgehensweise verstanden.

[70] Ifdef- Anweisungen sind bedingte Übersetzungen von bestimmten optionalen oder alternativen Programmstücken für verschiedene Systeme in der Programmiersprache C, z.B. im Embedded Bereich. Vgl. [Arno 2005] Folie 21 ff

klassen enthalten nur noch reine Logik. Das erhöht die Wartbarkeit und die **Wiederverwendbarkeit**. Diese Fachklassen werden in den restlichen Code „eingewoben".

Klassen selbst sind aber für eine komfortable Wiederverwendung oft zu klein, wenn sie nur eine Funktion abdecken. Deswegen haben sich gröbere Strukturen wie **Komponentenarchitekturen** oder **Plugins** bewährt. Ihre Granularität bestimmt sich nach Angaben der Performance und Skalierbarkeit, ihrem Installationsumfang und der Anzahl der Komponenten im System. Hier wird nur noch in Container, also Laufzeitumgebungen, die sich um die technischen Belange kümmern, und die Komponenten mit der reinen Berechnungslogik unterteilt. Gute Konfigurierbarkeit in den Komponenten gewährleistet die Verwendung für Systemfamilien.

Die Wiederverwendung findet nicht nur im Code, sondern auch in der Architektur und dem Design statt. Dies kann auf Designebene mit **Entwurfsmustern**, welche Designerfahrungen auf konzeptioneller Ebene beinhalten, und auf Architekturebene mit **Frameworks** geschehen. Allumfassend beschäftigt sich die **Generative Programmierung** mit dem Erstellen eines stabilen „Stecksystems" aus variablen Komponenten und der automatischen Ableitung über Generatoren, die z.B. auf **GenVoca** basieren. Persistente Einstellungen können im laufenden System z.B. in **XML** spezifiziert sein.

3.1.3 Integration der Konzepte in den Softwareentwicklungsprozess

Das Zusammenspiel dieser Methoden und Konzepte zielt auf die Entwicklung von Systemfamilien ab. Sie können nach folgendem Schema in die einzelnen Phasen des Softwareentwicklungsprozess eingegliedert werden, da ihre Aufgaben aufeinander aufbauen:[71]

[71] Vgl. [Völt 2000] S. 1 bis 8

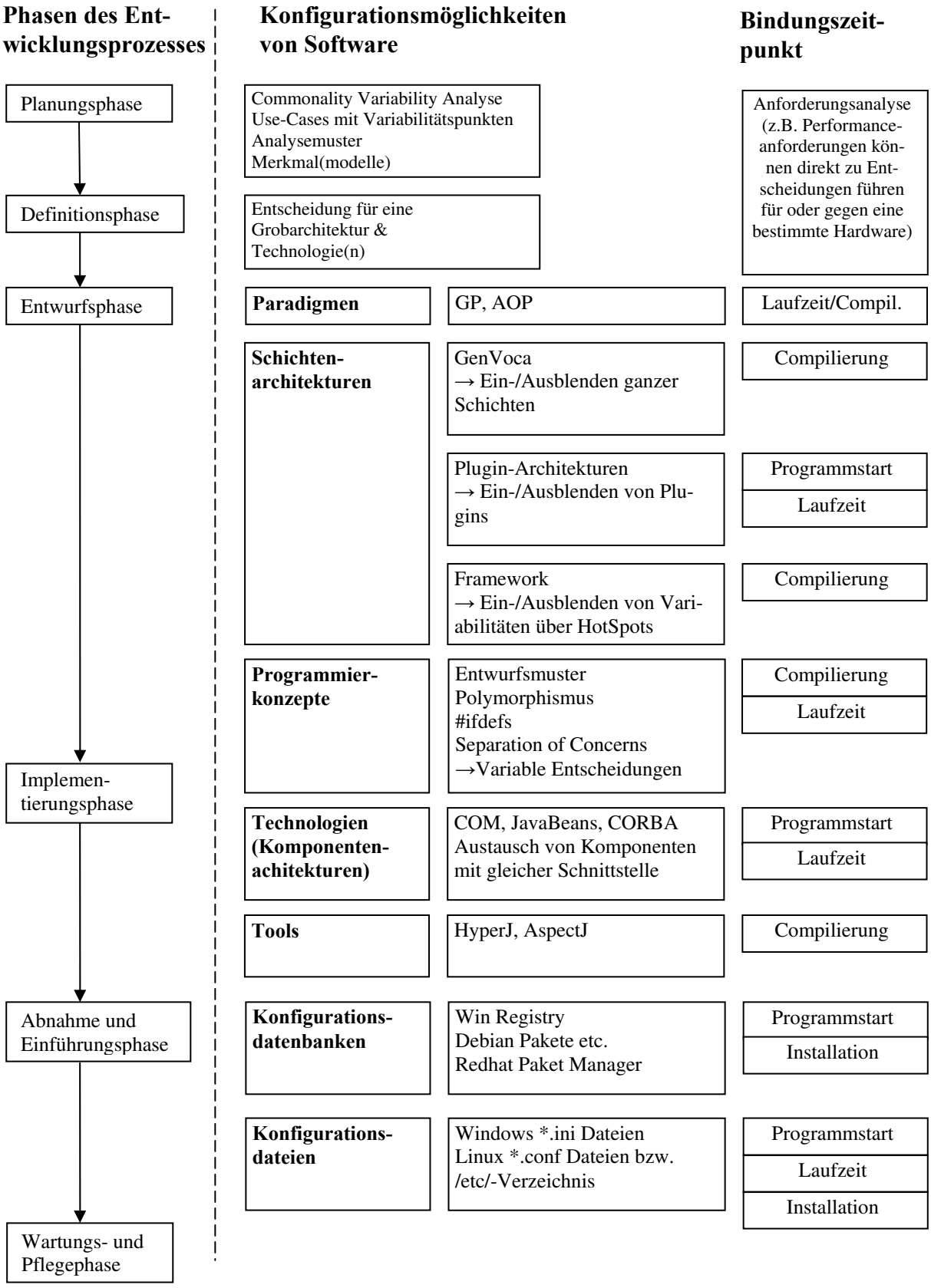

Phasen des Entwicklungsprozesses
Konfigurationsmöglichkeiten von Software
Bindungszeitpunkt
Planungsphase
Definitionsphase
Entwurfsphase
Implementierungsphase
Abnahme und Einführungsphase
Wartungs- und Pflegephase
Commonality Variability Analyse
Use-Cases mit Variabilitätspunkten
Analysemuster
Merkmal(modelle)
Anforderungsanalyse (z.B. Performanceanforderungen können direkt zu Entscheidungen führen für oder gegen eine bestimmte Hardware)
Entscheidung für eine Grobarchitektur & Technologie(n)
Paradigmen
GP, AOP
Laufzeit/Compil.
Schichtenarchitekturen
GenVoca
→ Ein-/Ausblenden ganzer Schichten
Compilierung
Plugin-Architekturen
→ Ein-/Ausblenden von Plugins
Programmstart
Laufzeit
Framework
→ Ein-/Ausblenden von Variabilitäten über HotSpots
Compilierung
Programmierkonzepte
Entwurfsmuster
Polymorphismus
#ifdefs
Separation of Concerns
→Variable Entscheidungen
Compilierung
Laufzeit
Technologien (Komponentenachitekturen)
COM, JavaBeans, CORBA
Austausch von Komponenten mit gleicher Schnittstelle
Programmstart
Laufzeit
Tools
HyperJ, AspectJ
Compilierung
Konfigurationsdatenbanken
Win Registry
Debian Pakete etc.
Redhat Paket Manager
Programmstart
Installation
Konfigurationsdateien
Windows *.ini Dateien
Linux *.conf Dateien bzw. /etc/-Verzeichnis
Programmstart
Laufzeit
Installation

Abbildung 2: Konfigurationsmöglichkeiten im Laufe des Entwicklungsprozesses[72]

Die Entwicklung von Systemfamilien beginnt in der Planungsphase[73] mit Vorüberlegungen wie der **Commonality Variability Analyse**, die sich mit der Untersuchung von gemeinsamen und veränderlichen Teilen beschäftigt, **Use-Cases**, die Anwendungsfälle modellhaft darstellen und **Merkmalmodellen**, wie z.B. FORE, die Anforderungen erheben. In der Definitionsphase werden die Entscheidungen und Anforderungen im Pflichtenheft spezifiziert. Die grobe umzusetzende Architektur wird festgelegt. Die Ergebnisse beeinflussen jegliches weitere Vorgehen grundlegend und bestimmen von Anfang an Leistungsmerkmale, die Aufteilung optionaler und obligatorischer Anforderungen und treffen erste Entscheidungen über nutzbare Hardware. Schließlich wird in der Entwurfsphase die vollständige Softwarearchitektur festgelegt. Dazu gehören auch Entscheidungen bezüglich der verwendenden Programmierparadigmen, wobei sich die **Generative Programmierung** als ein übergreifender Oberbegriff bei der Entwicklung von Systemfamilien entwickelt hat, welches alle nachfolgenden Paradigmen wie die **Aspektorientierte Programmierung**, Architekturen, Konzepte und Technologien verwendet. Schichtenarchitekturen wie **GenVoca**, **Plugins** und **Frameworks** orientieren sich an der Strukturierung durch Ausblenden bestimmter Schichten, Funktionsbereiche oder variabler Programmteile. Dabei kann der Bindungszeitpunkt, also das Festlegen der variablen Teile von Familienmitgliedern frühestens zur Compilerzeit, spätestes zum Programmstart, wie z.B. bei der Demo-CD in Kapitel 4 oder speziell bei Plugins auch noch während der Laufzeit geschehen. Bei der VDR-Software können z.B. per OSD während der Programmausführung Plugins aufgerufen und in das Programm eingebunden werden. Zum Entwurf und der anschließenden Implementierung gehören dazu Programmablaufsentscheidungen, die über Konzepte wie **Polymorphismus**, **Entwurfsmuster**, einfache **ifdef-Anweisungen** oder der grundlegenden Denkweise des **Separation of Concerns** erfolgen, variable Programmabschnitte werden bei diesen Konzepten zur Compilerzeit realisiert. Entwurf und Implementierung des Programms können dabei von Komponentenarchitekturen wie **COM**, **CORBA** und **JavaBeans** vereinfacht werden, da sie bereits fertige Komponenten bieten, die in ein Programm eingebunden werden können. Die Bindung variabler Teile erfolgt hier beim Programmstart oder auch dynamisch während der Laufzeit, wenn z.B. **ActiveX** Elemente in einem Browser installiert werden. Zur Implementierungszeit wird die Realisierung der Softwarearchitektur von Tools wie **HyperJ** von IBM oder **AspectJ** des Eclipse-Projektes[74] unterstützt. Hier entwickelte,

[72] Vgl. [Balz 2001] S. 1 ff und [Arno 2005] Folie 54 ff

[73] Die Phaseneinordnung der einzelnen Themen erfolgte anhand von [Balz 2001] S. 1 ff.

[74] Eclipse ist eine freie, aber plattformabhängige Software-Entwicklungsumgebung (IDE), primär für Java, die mit Hilfe von Plugins aller Art erweitert werden kann, ursprünglich von IBM entwickelt und dann freigegeben wurde und nun von freien Entwicklergemeinden vieler großer Firmen stetig vorangetrieben wird. Vgl. [Wiki 2005] Eclipse

spezifische Anteile eines Familienmitglieds werden zur Compilerzeit festgelegt. Nach Einführung der Software können über **Konfigurationsdatenbanken und -dateien** einzelne spezifische Einstellungen zur Installation wie Pfadangaben, zum Programmstart wie Servereinstellungen wie z.B. beim Apache-Server und während der Laufzeit wie z.B. Farb- und Bildgrößeneinstellungen des Bildschirms vorgenommen werden. Stellvertretende Themen der einzelnen Bereiche werden im Folgenden näher untersucht. Die Bewertung erfolgt in Anlehnung an die geforderten Merkmale der DIN ISO 9126, in der Softwarequalität nach den Kriterien[75] Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit bewertet wird.

Die genannten Punkte in der jeweiligen Bewertung wurden der referenzierten Literatur der Unterkapitel entnommen, und können nicht als allgemeingültig angesehen werden, da die Entscheidung zu bestimmten Werkzeugen[76], Paradigmen oder Architekturen projektspezifisch getroffen werden muss und hier Abhängigkeiten und erfahrungsgemäß üblich angewendete Kombinationen von komplexen und weniger komplexen Mechanismen bestehen, die sich für spezifische Probleme besonders gut eignen,[77] denn „Während früher davon ausgegangen wurde, dass Vorgehensmodelle zur Softwareentwicklung weitestgehend neutral in Bezug auf die Art der mit ihnen zu realisierenden Softwaresysteme sind, kann bei neueren Methoden ganz klar der Zusammenhang zwischen der Wahl der Methoden und der damit verbundenen Vorstellung über die Art des zu erstellenden Systems gesehen werden.“[78]

Für die Unterstützung der Auswahl von Implementierungshilfen gibt es z.B. von der FH Mannheim Bestrebungen, mittels eines Kataloges aus Kiviat-Diagrammen[79] und Beschreibungskarten geeignete Vorschläge zu liefern.[80]

3.2 Softwareengineeringkonzepte zur Konfiguration von Systemfamilien

Grundlegend wird durch die Konfiguration der Softwarearchitektur auch die Beschaffenheit und Erweiterbarkeit der Software nach der Implementierung bestimmt. Deswegen folgen nach der Einleitung zur Entwicklung von Systemfamilien die genaue Analyse der bereits im vorhergehenden Kapitel erwähnten Paradigmen, Architekturen und Hilfswerkzeuge, die bei der Erstellung von Familien in den ersten Phasen verwendet werden.

[75] Zitiert nach [Balz 2001] S. 1113

[76] Zur Definition eines Werkzeuges bzw. Tools vgl. [Balz 2001] S. 38: Ein Werkzeug dient der automatisierten Unterstützung von Methoden.

[77] Vgl. [Arno 2005] Folie 67 ff

[78] Vgl. [Stru 2005] Prozess

[79] Ein Kiviat-Diagramm stellt die Ausprägungen mehrerer abhängiger Merkmale in einer Art Spinnennetz dar.

[80] Vgl. [Arno 2005] Folie 56 ff

3.2.1 Einleitung in die Entwicklung von Systemfamilien

Softwareentwicklung stellt Ansprüche im Bereich neuer Anwendungsgebiete, komplexen Aufgaben und wachsenden Anforderungen, im Bereich Qualität mit schwerwiegenden Konsequenzen von Software-Fehlern, Akzeptanzkriterien, unterschiedlich definierter Softwarequalität und steigendem Bedarf an neuen Systemen und Varianten, wachsender Nachfrage und der ungenügenden Funktionsabdeckung von Standardsoftware. Dazu kommt die Belastung durch die hohe Lebensdauer von Altsystemen und deren Bindung von Entwicklerkapazitäten, was bei neuen Systemen berücksichtigt werden muss. Hier greifen bekannte Systeme der Wiederverwendung von Software mittels komponentenbasierter Softwareentwicklung ein. Ziel ist die kostengünstige, anforderungsspezifische Generierung einer großen Zahl qualitativ hochwertiger Systemvarianten, die weitgehend automatisiert werden kann.[81]

Im Rahmen dieser Diplomarbeit werden Komponenten wie folgt definiert:

Definition: Komponenten[82]
Eine **Komponente** ist ein „Bestandteil eines Ganzen“, in der Informatik ein abgeschlossener Teil eines Softwareprogramms bestehend aus einer Folge von Verarbeitungsschritten und Datenstrukturen. Der stark kohäsive Inhalt enthält anwendungsorientierte, semantisch zusammengehörende Funktionalitäten wie wiederverwendbare Berechnungen oder Bearbeitungsschritte von Daten, wobei Schnittstellen diese nach außen zur Verfügung stellen. Über die Schnittstellen werden ausschließlich Daten übernommen oder abgegeben, der Inhalt der Komponente bleibt gekapselt. Sie müssen weitgehend redundanzarm, wieder verwendbar, frei kombinierbar und offen konzipiert sein, um Änderungen vornehmen zu können. **Module** können ein oder mehrere Komponenten enthalten. **Plugins** sind den Komponenten ähnlich, stellen aber oft autonomere Teile dar.

Tabelle 12: Definition einer Komponente

Ulrich W. Eisenecker et al benennen die Vorteile der Systemfamilien wie folgt[83]: „Die Softwareentwicklung ist überwiegend auf die Erstellung einzelner Systeme ausgerichtet, obwohl überall auf die Notwendigkeit von Wiederverwendung hingewiesen wird. Doch wie soll die Entwicklung von Einzelstücken etwas Wiederverwendbares hervorbringen? Zwar bringen Frameworks, Entwurfsmuster und Softwarekomponenten in dieser Hinsicht Besserung, aber keinen Durchbruch. Einen wesentlichen Fortschritt verspricht hingegen die Einwicklung von Systemfamilien. Sie ermöglicht die anforderungsspezifische Fertigung einer großen Zahl von

[81] Vgl. [Eise 2003-3] S. 1 ff
[82] Vgl. [Stre 2004] S. 11ff, [Wiki 2005] Komponente, [Eise 2003-2] S. 4 und [Balz 2001] S. 856
[83] Vgl. [Eise 2002] S. 1

Systemvarianten, und die kann weitgehend automatisiert werden. Dies erlaubt die drastische Senkung von Entwicklungszeiten und -kosten sowie wesentliche Verbesserungen der Qualität."

In dieser Diplomarbeit wird eine Systemfamilie wie folgt definiert:

Definition: Systemfamilien[84]
Eine Systemfamilie ist ein Softwaresystem, welches aus einer **Referenzarchitektur**, der **Dokumentation** und verschiedenen optionalen und obligatorischen **Komponenten** besteht, die eine automatisierbare Ableitung eines Familienmitgliedes erlauben. Die planbare und umfassende Verwendung von Komponenten in einer Anwendungsdomäne steht im Vordergrund als großer zeitlicher und wirtschaftlicher Faktor. Auch der Begriff „**Produktlinie**" wird in der Literatur gleichbedeutend verwendet

Tabelle 13: Definition von Systemfamilien

Die technische Seite einer Systemfamilie und die Ableitung der Familienmitglieder sind der Abbildung 3 zu entnehmen. Die Konfiguration einer Applikation wird dabei automatisch anhand der bestehenden Komponenten und der Referenzstruktur mittels eines Generators hergestellt.

Die grafische Darstellung kann der folgenden Abbildung entnommen werden:

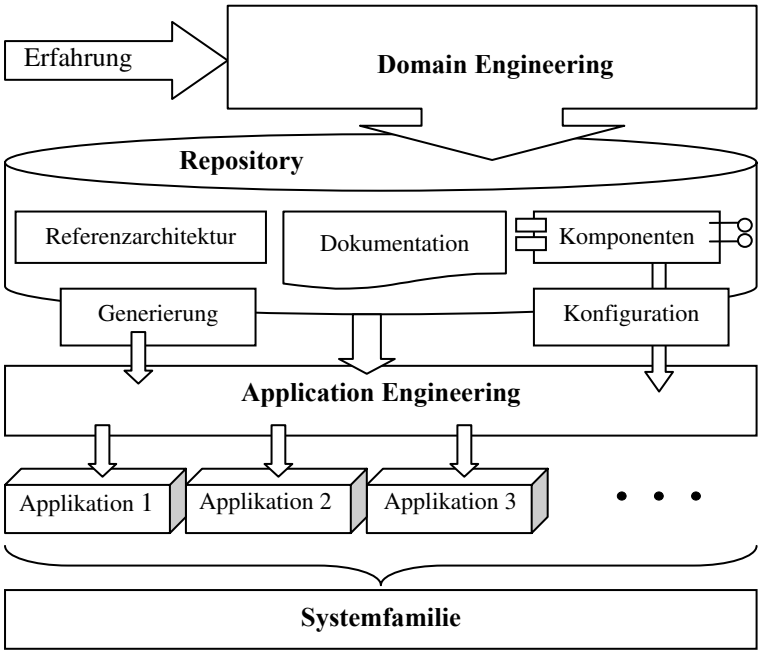


Abbildung 3: Das Entwurfsprinzip der Systemfamilie[85]

[84] Vgl. [Stre 2004] S. III

3.2.2 Programmierparadigmen

3.2.2.1 Generative Programmierung

Ausgehend von den bereits beschriebenen Ansprüchen von Softwareentwicklung modelliert die ganzheitliche Generative Programmierung Softwaresysteme so, dass „ausgehend von einer Anforderungsspezifikation mittels Konfigurationswissen aus elementaren, wiederverwendbaren Implementierungskomponenten ein hochangepasstes und optimiertes Zwischen- oder Endprodukt nach Bedarf automatisch erzeugt werden kann“.[86] Dazu verwendet die Generative Programmierung aufeinander abgestimmte Methoden und Techniken zur Entwicklung von Softwaresystemfamilien. Wesentliche Bestandteile des Generativen Programmierens sind die merkmalbasierte Modellierung von Gemeinsamkeiten und Unterschieden der Familienmitglieder und die Entwicklung von Sprachmitteln zur Beschreibung einzelner Systeme sowie die Entwicklung von Generatoren, die diese aufgrund ermittelter Anforderungen weitgehend automatisch erzeugen.[87] Die Vorgehensweise des Generativen Programmierens ist in der folgenden Abbildung dargestellt:

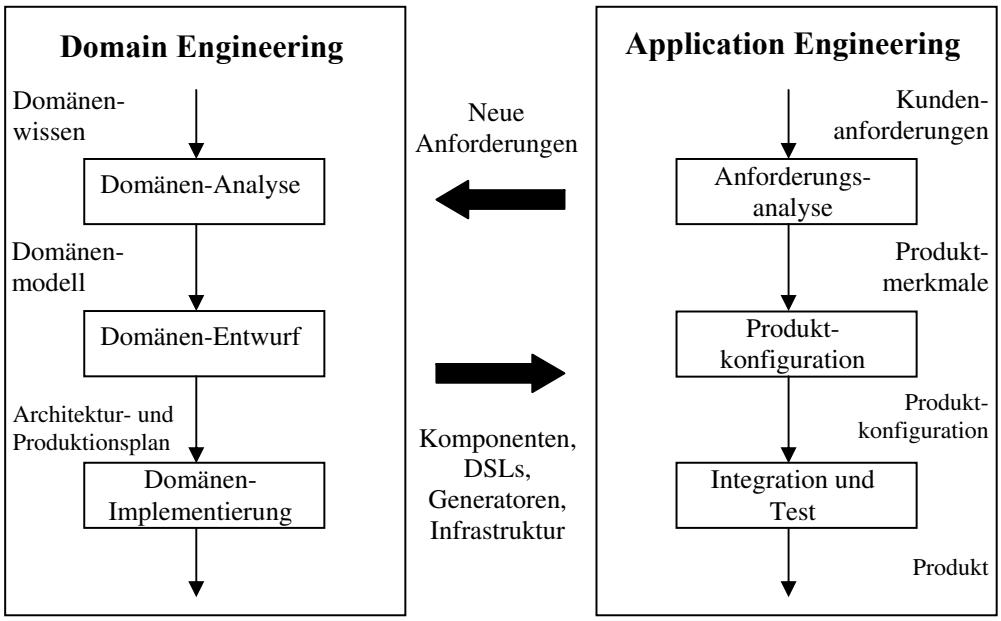


Abbildung 4: Systemfamilienentwicklung nach Czarnecki[88]

Die grundlegenden Bestandteile des Generativen Programmierens sind das **Domain Engineering** und das **Application Engineering**. Im Domain Engineering wird zuerst das Umfeld ana-

[85] Vgl. [Alex 2004] Folie 4, [Rieb 2000]
[86] Vgl. [Czar 2001-2] Folie 8
[87] Vgl. [Eise 2003-2] S. 1 ff
[88] Vgl. [Czar 2001-2] Folie 7

lysiert: Welche Personen mit welchen Interessen sind beteiligt, welche Anforderungen werden gestellt. Im wichtigsten Teil, dem Merkmalmodell wird zusammenfasst, welche Beziehungen die Merkmale der Familie haben. Ein solches Modell ist z.B. das in Kapitel 2.1.2 vorgestellte FORE-Modell. Zur Konsistenzüberprüfung des Merkmalmodells gibt es derzeit an der TU Ilmenau ein Forschungsprojekt. Für die pluginbasierte Software Eclipse gibt es bereits ein solches Überprüfungsplugin namens pure::variants[89].

Dazu müssen Tabellen und Bibliotheken entstehen, die das entsprechende Wissen über Abhängigkeiten, Wechselwirkungen, Beschreibungen, Herkunft und Verwendung von Merkmalen, sowie die Standardvorgaben, Bindungszeitpunkte etc. protokollieren. Daraus lässt sich ableiten, welche Funktionen im Basissystem zwingend dazugehören müssen und welche sich optional in Familienmitgliedern realisieren lassen. Dafür muss untersucht werden, welche relevanten Bausteine sich in entsprechenden Kategorien verschiedener Implementierungen zusammenfassen lassen und welche Abhängigkeiten bestehen.

Grundstein dieses Vorgehens ist das **Generative Domain Model**, welches den Problemraum, also die Fachbereiche und die spezifischen Anforderungen enthält, den Lösungsraum, der alle Komponenten des Systems beschreibt, und das Konfigurationswissen, welches den Bauplan für die Komponenten zur Lösung konkreter Problembeschreibungen liefert.

Diese Ergebnisse fließen in den Entwurf der flexiblen Softwarearchitektur für die komplette Anwendungsfamilie und werden dann von wieder verwendbaren Komponenten, einer dafür entwickelten domänenspezifischen Beschreibungssprache, der Domain Specific Language, und von Konfigurationsgeneratoren implementiert. Die Ergebnisse des Domain Engineering sind eine Art Katalog für mögliche automatische Ableitungen ähnlich einer Angebotsbroschüre von Automobilherstellern. Dieser Katalog ist die Grundlage für die Entwicklung echter Kundenanwendungen im Application Engineering. Hier werden die spezifischen Kundenanforderungen erhoben und mittels der bestehenden Komponenten, Sprachen und Generatoren mit Hilfe von Templates und dem Konfigurationswissen zusammengesetzt. Anforderungen, die hier neu erhoben werden, fließen über das nötige Konfigurationsmanagement wieder in das Domain Engineering ein.[90]

[89] Das vollständig modellbasierte kommerzielle Plugin der Firma pure-systems GmbH unterstützt die Entwicklung und Ableitung von Systemfamilien, indem es Merkmalmodelle für diese überprüft und mittels eines eigenen Transformators eigenständig in Familienmitglieder mit Hilfe vorhandener Komponenten übersetzt. Vgl. [Pure 2005]

[90] Vgl. [Eise 2003] Technik und [Czar 2001-02] Folie 7 ff

Das Vorgehen des Generativen Programmierens hat folgende Vor- und Nachteile:

Bewertung der Generative Programmierung[91]	
Vorteile	**Nachteile**
• Ganzheitliches Konzept • Fokussiert Softwareerstellung auf „Knopfdruck“ • Verringert den Rückstand der Softwareentwicklung gegenüber der Industrie • Komponentenbasierte Denkweise verbessert Wartung von Einzelteilen und die parallele Entwicklung • Kürzere Entwicklungszeit bei höherer Produktivität • Zukunftsweisendes Vorgehen • Fehlerreduzierung durch Automatisierung, Fehlerbehebung direkt im Lösungsraum • Aufteilung in einzelne Abschnitte erleichtert Projektmanagement mit kürzeren Analyse- und Einwicklungszeiten von Varianten • Plattform- und Programmiersprachenunabhängigkeit • Im bestehenden System übernimmt der Generator die Codegenerierung, Programmierer werden entlastet	• Das Domain Engineering ist ein fehleranfälliger Prozess, hier vergessene Features führen zu eingeschränkten Entwicklungsmöglichkeiten • Überarbeitungen sind hier schwierig und langwierig • Der Aufwand ist enorm, es können Jahre bis zum ersten Produkt vergehen • Die Vielschichtigkeit erfordert sehr erfahrene Teammitglieder • Integrationsprobleme bei Verwendung verschiedener Programmiersprachen für die Ableitung und damit verbundene Performanceprobleme

Tabelle 14: Bewertung der Generativen Programmierung

[91] Vgl. [Tamm 2004] S. 1 ff, [Benz 2003] S. 1 ff und [Brüg 2005]

3.2.2.2 Aspektorientierte Programmierung

Schnelle Entwicklungszyklen durch sich häufig ändernde Anforderungen erfordern flexible Softwarearchitekturen. Hier setzt das 1997 veröffentlichte Paradigma der Aspektorientierte Programmierung an. Diese soll die objektorientierte Programmierung erweitern, wird aber auch als eigenständiges Paradigma[92] welches auch COM, CORBA oder Entwurfsmuster nutzt, verstanden.

Hauptidee der Aspektorientierten Programmierung ist die vollständig Realisierung des **Separation of Concerns**, die fachliche Trennung wichtiger Sichtweisen oder Teile in der Programmierung, denn viele Funktionalitäten sind in der objektorientierten Programmierung auf diverse Klassen verteilt, so genannte **Cross-Cutting-Concerns**. Dazu bietet die Aspektorientierte Programmierung Methoden und Techniken an, um funktionale Komponenten und Aspekte orthogonal zur Programmsemantik zu differenzieren, und diese wieder in eine Gesamtkomposition zurückzuführen. Die Wiederverwendbarkeit der Teile steht dabei im Vordergrund. Aspekte verkörpern hierbei die über den Klassen verstreuten Codefragmente von Funktionen, die das Softwaresystem an diversen Stellen beeinflussen. Beispiele hierfür sind Rechteüberprüfungen, Persistenz, Synchronisation von Daten, Objektinteraktionen und Parameterübergaben. Sie verkörpern einerseits die Anforderungen und kapseln sie gleichzeitig als Programmkonstrukt.[93] Sie können Attribute, Kostanten, Member-Typen und Funktionen enthalten, oder auch Algorithmenteile ersetzen und später aufrufen. Die Aspekte werden in eigenen Klassen oder anderen Sprachfragmenten ausgelagert. Verbindungspunkte, sogenannte **Join Points**, verbinden sie mit dem Rest des Programms. Ein **Weaver** kontrolliert, welche Klassen von welchen Aspekten angesprochen werden sollen und „verwebt“ sie untereinander. Der Vorteil der Aspektorientierung liegt darin, dass Aspekte auch nach der Kompilierung in Komponentenarchitekturen eingepflegt werden können, da die Aspektorientierte Programmierung hilft, die entscheidenden Programmteile, die Concerns, zu erkennen, zu erweitern und wieder zu verwenden. Dazu müssen (vgl. Abbildung 5)[94]

1. Aspekte in einem objektorientierten Programm (grau hinterlegt) erkannt werden
2. Effiziente Mechanismen für das Zusammenstellen und Ausdrücken der Aspekte gefunden werden
3. Und in dem objektorientierten Programm verwoben werden

[92] Zur Definition eines Paradigmas vgl. [Balz 2001] S. 40: Ein Paradigma ist ein der Programmiertechnik zugrunde liegendes Prinzip bzw. eine Denkweise.
[93] Vgl. [Czar 2000] S. 264 ff
[94] Vgl. [Czar 1998] S. 182 ff

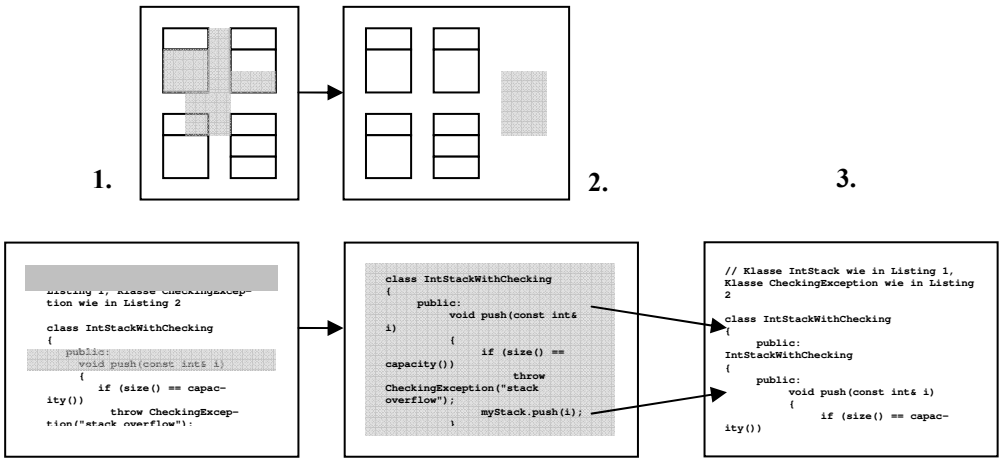


Abbildung 5: Vorgehensweise der Aspektorientieren Programmierung

Die Vor- und Nachteile seien hier zusammengefasst:

Bewertung der Aspektorientierten Programmierung[95]	
Vorteile	**Nachteile**
• Verbessert die Lesbarkeit und Verständlichkeit des Codes, da Concerns im Mittelpunkt stehen und Aspekte sie kapseln • Aspekte fördern die saubere Programmierung, da durch ständige Änderungen in der klassischen objektorientierten Programmierung die Übersichtlichkeit verloren geht • Vielseitig anwendbarer Ansatz • Programmerweiterungen leicht realisierbar • Verbessert die Wartbarkeit, Pflege und Konfiguration eines Programms, da sich fachliche Änderungen nicht an diversen, sondern nur noch an einer Stelle niederschlagen • Die Wiederverwendbarkeit von Komponenten wird verbessert • Späte Designentscheidungen sind möglich • Aufgaben im Softwareentwicklungsteam können entlang der Aspekte sinnvoller verteilt werden	• Bei aufwändigen Projekten kann eine Überfrachtung sich wiederum negativ auf die Verständlichkeit auswirken • Die vorhergehende Analyse hat eine anfängliche Mehrarbeit zur Folge • Nicht alle Aspekte können immer adäquat umgesetzt werden • Da die Aspektorientierte Programmierung noch entwickelt wird, gibt es noch keine allgemeingültigen Kompositionsregeln für das Zusammenfügen der Komponenten • Die Semantik erschließt sich aus Einzelmodulen nicht mehr • Testen und Verifizieren gestalten sich schwierig • Hohe Kopplung zwischen dem Code und den Aspekten

Tabelle 15: Bewertung der Aspektorientierten Programmierung

[95] Vgl. [Czar 2000] S. 267 ff, [Czar 2001] und [Arno 2005] Folie 40 ff

Einen ähnlichen Focus haben die Subjektorientierte Programmierung, Compostion Filters oder die Adaptive Programmierung.[96]

3.2.3 Schichtenarchitekturen

3.2.3.1 GenVoca

Die GenVoca Methode entstand 1988 aus den beiden Projekten Genesis und Avoca von Don Batory und O'Malley, die sich mit der Anwendung von Data Container Libraries beschäftigten. Im Bereich der Generativen Programmierung wird GenVoca als formale Beschreibung von Schichtenarchitekturen und Grammatik für die Entwicklung eines Generators zum Austausch von Komponenten mit gleicher Schnittstelle verwendet, und fokussiert ebenso Wiederverwendung und Systemfamilien. Ein Generator hat das Ziel, Softwaresysteme automatisch zu erstellen. Don Batory dazu: „Given a GenVoca model, we can create a family of applications by composing features".[97]

Er schließt die Lücke zwischen der Systembeschreibung und der Implementierung des Systems, und muss spezifische Anforderungen an Geschwindigkeit, Speicherbelegung, Ressourcenverwaltung etc. erfüllen. GenVoca löst das Problem der Erstellung eines Generators mittels eines kompositionellen Ansatzes als Schichten-orientiertes Modell. Jeder abstrakte Datentyp oder eine Eigenschaft wird in einer eigenen Schicht dargestellt. Jede Schicht enthält dabei eine Anzahl von Klassen, die die untergeordnete Schicht weiter mit Klassen und Methoden verfeinern. Die Klassen verdeutlichen parametrisierbare Komponenten, die sich aus Komponentensicht in einer Baumstruktur darstellen lassen. Das Konfigurationswissen aus der Domänenanalyse wird dabei Schicht für Schicht weiter getragen.[98] Ein fertiger Generator muss im Bereich der Systemfamilienentwicklung eine Anforderungsspezifikation prüfen, vervollständigen, optimieren und die Implementierung generieren. Dies können fertige Softwareprodukte und ebenso Halbprodukte sein.[99]

[96] Vgl. [Czar 2000] S. 254 ff
[97] Vgl. [Bato 2005] S. 1 ff
[98] Vgl. [Czar 1998] S. 119 ff und 134 ff
[99] Vgl. [Tamm 2004] S. 2

Ein beispielhafter Ausschnitt verdeutlicht dies: [100]

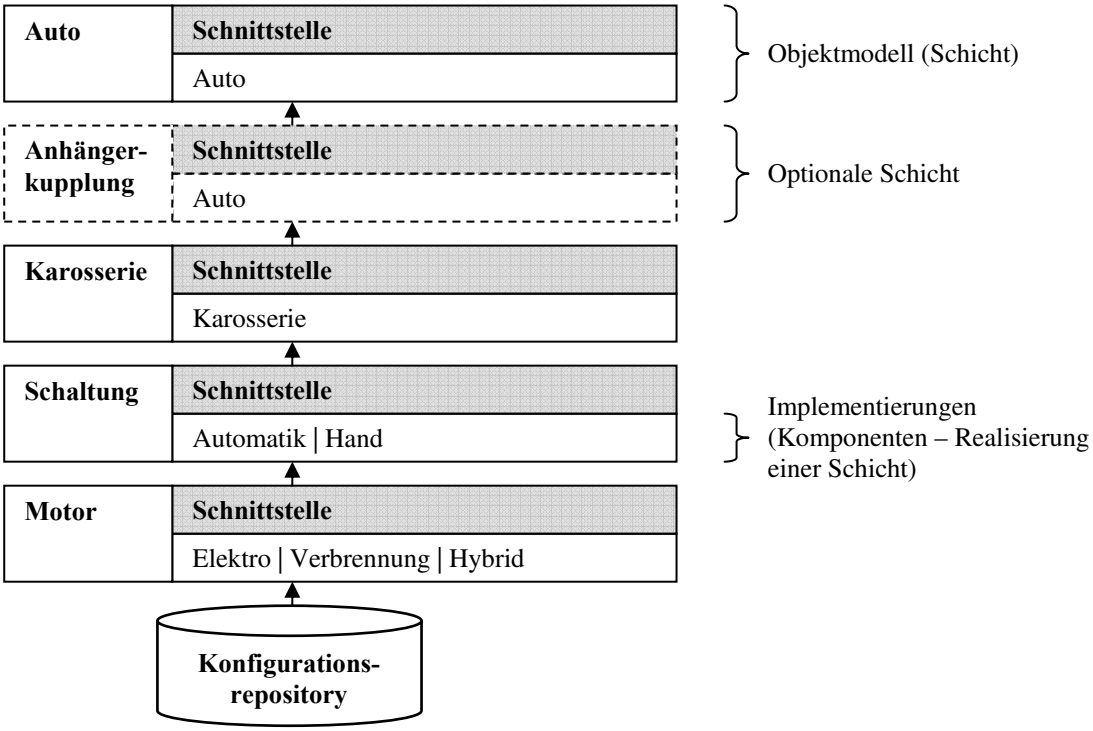


Abbildung 6: GenVoca-Schichten im Beispiel

Die Komponenten einer Schicht werden in einer Kategorie (realm) zusammengefasst. Eine Kategorie kann so als eine Bibliothek von austauschbaren Komponenten verstanden werden, die über eine einheitliche Schnittstelle nach außen hin repräsentiert wird. Über Parameter wird dann der oberen Schicht übergeben, welche Komponente der untergeordneten Schicht bei der Realisierung verwendet werden soll. Der Aufbau der Schichten und deren Kategorien wird in eine kontextfreie GenVoca Grammatik überführt, die mit Hilfe der Komponentenkategorien und Entwurfsregeln aus dem Konfigurationswissen besagt, welche Kombinationen an obligatorischen oder wahlpflichtigen Teilen möglich sind. Jeder Ausdruck entlang der Grammatik ist ein realisierbares Familienmitglied.[101]

Auto:	Auto [Karossiere_mit_opt_AK]
Karosserie_mit_Opt_AK:	Karosserie_mit_AK[Fertige_Karosserie] |Karosserie[Schaltung_mit_Motor]
Fertige_Karosserie:	Karosserie[Schaltung_mit_Motor]
Schaltung_mit_Motor:	Handschaltung[Motor] | Automatikschaltung[Motor]
Motor:	Verbrennungsmotor | Eletromotor | Hybridmotor

Abbildung 7: GenVoca Grammatik im Beispiel

[100] Vgl. [Brüg 2005] Kapitel 2 ff und [Eise 2002-2] Folie 54 ff

[101] Vgl. [Brüg 2005] Kapitel 2 ff und [Eise 2002-2] Folie 1 ff

Der in der Grammatik hinterlegte Aufbau wird mittels einer geeigneten Sprache als Templates umgesetzt. Ein solcher entwickelter Generator kann nun alle Schichten Schritt für Schritt durchlaufen und Mitglieder der beschriebenen Systemfamilie entwickeln.

Bewertung des GenVoca-Modells[102]	
Vorteile	**Nachteile**
• Einfache verständliche Beschreibung mittels Grammatik • Schnelle und günstige Produktableitung bei einem eingerichteten Generator • Fördert hohe Codequalität mit wenig Fehlern • Erstellung neuer Produktvarianten sowie Halbprodukten möglich	• Die Erstellung eines Generators und der Grammatik bedeutet einen enorm hohen Aufwand • Spätere Anpassung bedeuten auch ein Anpassen der Grammatik und des Generators mit hohem Zeit- und Kostenbedarf • Erschwerte Wartbarkeit

Tabelle 16: Bewertung des GenVoca-Modells

3.2.3.2 Plugins

Pluginarchitekturen sind weit verbreitet und seit 1987 bekannt[103], denn jeder Gerätetreiber ist auch ein Plugin. Prominente Beispiele sind die Adobe Produkte, alle möglichen Plugins für Browser wie das Java-, das Flash- oder das Realplayer-Plugin und die Entwicklerplattform Eclipse sowie die hier verwendete VDR-Software.

Plugins sind dabei funktional selbständige Ergänzungen zu einer Applikation, die eine fachliche Funktion vollständig abdecken, ohne dass andere Komponenten informiert werden müssten. Die Applikation selbst ist für die Schnittstellen zuständig. Plugins sind damit unabhängigere Bausteine als Komponenten. Zudem sind sie ausschließlich für eine Anwendung geschrieben und müssen in ein vorhandenes System integriert werden, arbeiten also auch nur mit dessen Anwendungsdaten.

Plugins sind anders als Komponenten so tatsächlich als optional zu verstehen und erweitern eine bereits funktionierende Software um bestimmte Funktionen oder Funktionsgruppen. Plugins werden dabei in das bestehende System eingeklinkt und enthalten ihre eigene Konfiguration. Sie können jedoch nicht ohne die integrative Anwendung genutzt werden. Der Hauptunterschied zu Komponenten liegt aber in der Entstehung. Komponentenarchitekturen gliedern eine Anwendung in die Berechnungslogik und die Infrastruktur, also die technischen Belange,

[102] [Arno 2005] Folie 51 ff
[103] Vgl. [Wiki 2005] Plugin

hier gilt: Logik und Container werden getrennt. Plugins dagegen gliedern die gesamte Anwendung in gleiche funktionale Anschnitte und „stecken sie wieder ineinander“, daher der Name Plugins, hier gilt: Anwendungen werden durch Plugins getrennt.

Ihre Anwendung kann wie im Browserbeispiel für den Nutzer unsichtbar bleiben, wenn dieser ein Plugin selbständig herunterlädt und installiert; im Adobebeispiel reserviert sich jedes Plugin einen Bereich im Menü. Durch ihren Aufbau sind innovative Ideen möglich, Eclipse bietet so z.B. Plugins für Codegeneratoren, Datenbankbrowser, Wikis etc. Plugins können sich auch gegenseitig bedingen und geben das zum Installationszeitpunkt an. Bei der Installation müssen sich die Plugins bei der Anwendung registrieren, um deren Funktionen nutzen zu können. Der Aufbau einer Pluginhierarchie kann wie folgt dargestellt werden:[104]

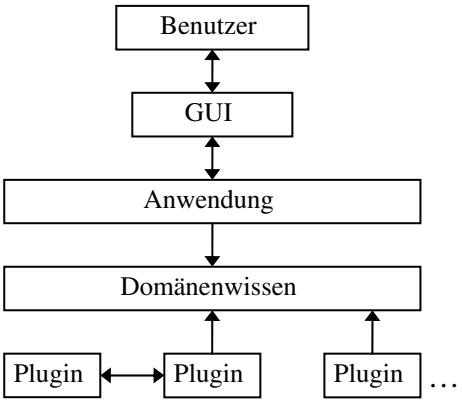


Abbildung 8: Aufbau einer Pluginarchitektur

Das Domänenwissen beinhaltet dabei die Informationen, die zur Installation, Integration und Registrierung eines neuen Plugins nötig sind, sowie Informationen zum Lebenszyklus und zu den Diensten eines Plugins, ohne die Funktionslogik der Plugins tatsächlich zu kennen.

Pluginarchitekturen bieten folgende Vor- und Nachteile:

[104] Vgl. [Völt 2000] S. 1 bis 8

Bewertung von Pluginarchitekturen[105]	
Vorteile	**Nachteile**
• Sehr leichte Erweiterung • Plugins sind unabhängig von der Gesamtanwendung • Schnellere, parallele Entwicklung im Projektteam möglich • Austauschbarkeit bei veränderten Anforderungen sehr flexibel • Ermöglicht eine nahtlose Integration von Funktionen ohne komplexe Infrastruktur • Leichte Wartbarkeit • Separation of Concern in Reinform • Einheitliche Schnittstellen für alle Plugins • Ermöglicht die Entwicklung innovativer Ideen • Optimale Grundlage für die Entwicklung von Systemfamilien • Plugins können einzeln vertrieben werden	• Installation wird immer aufwändiger, da jedes Plugin einzeln integriert wird • Updates müssen für jede Funktionalität einzeln eingespielt werden • Sehr große Pluginarchitekturen sind schwerer zu managen • Ausschließlich für eine Anwendung konzipiert • Die Erstellung des Domänenwissens ist schwieriger zu bewerkstelligen • Entwicklung einer einheitlichen Schnittstelle für alle Belange ist ein komplexeres Thema • Automatische Installation von Browserplugins können ein Sicherheitsrisiko bedeuten

Tabelle 17: Bewertung von Pluginarchitekturen

3.2.4 Programmierkonzepte

3.2.4.1 Entwurfsmuster[106]

Entwurfsmuster „sind Beschreibungen zusammenarbeitender Objekte und Klassen, die maßgeschneidert sind, um ein allgemeines Entwurfsproblem in einem bestimmten Kontext zu lösen. Ein Entwurfsmuster benennt, abstrahiert und identifiziert die relevanten Aspekte einer allgemeinen Entwurfsstruktur“[107], hilft bei der Komponentenentwicklung und stellt wiederum selbst universelle Komponenten dar, die flexibel wiederverwendet werden können. Sie bieten einfache und elegante Lösungen für spezifische Probleme des objektorientierten Softwareentwurfs, indem sie einfache und konzentrierte Problemlösungen darstellen.

Entwurfsmuster sind durch die Überlegung entstanden, dass sich in jeder gut strukturierten, objektorientierten Architektur Muster finden lassen müssen. Daraus resultieren weniger um-

[105] Vgl. [Völt 2004] S. 1 bis 10
[106] Vgl. [Phil 2003] S. 42 ff
[107] Vgl. [Gamm 1996] S. 3 ff

fangreiche und leichter verständliche Architekturen, die bereits erfolgreich Anforderungen umgesetzt haben, und konzentrieren das Wissen und die Erfahrungen der Programmierer so, dass immer wiederkehrende Probleme nicht neu gelöst werden müssen. Sie bestehen objektorientiert aus Objekten, Klassen, Schnittstellen und Vererbungshierarchien.

Der Katalog nach Gamma enthält 23 Muster in 3 verschiedenen Kategorien[108]: Erzeugungsmuster, Strukturmuster und Verhaltensmuster, die sich jedoch häufig ähneln. Sie beinhalten grundlegende Elemente wie

Mustername: Als knappe Bezeichnung des Problems und Verdeutlichung des Vokabulars.

Problemabschnitt: Als Problem- und Kontextbeschreibung mit Bedingungen für den Einsatz eines Musters (Motivation, Zweck, Grundprinzip, Fragestellungen, Anwendungsdiagramme).

Lösungsabschnitt: Als abstrakte Beschreibung der Elemente, Beziehungen, Zuständigkeiten und Interaktionen und Anordnungen der Elemente des Entwurfs als eine Schablone, hier gehört auch ein Beispielcode dazu.

Konsequenzabschnitt: Als Beschreibung der Vor- und Nachteile bei Verwendung, z.B. Speicherplatzverbrauch, Ausführungszeit, Sprach- und Implementierungsaspekte, Einfluss auf Flexibilität, Erweiterbarkeit und Portabilität eines Systems, Variantenmöglichkeiten, Fallen, Tipps und Techniken für die Implementierung, verwandte Muster und bekannte Anwendungsfälle und welche Muster zusammen verwendet werden können.

Beispielhaft sei hier das Strukturmuster für einen Adapter bzw. Wrapper erläutert[109]. Das Muster dient der Schnittstellenimplementierung zwischen inkompatiblen Klassen ohne dass die Klassen selbst geändert werden müssen. So ist eine Wiederverwendung von Klassen einer Klassenbibliothek möglich, was sonst nicht der Fall wäre. Die Vorgehensweise ist Abbildung 9 zu entnehmen:

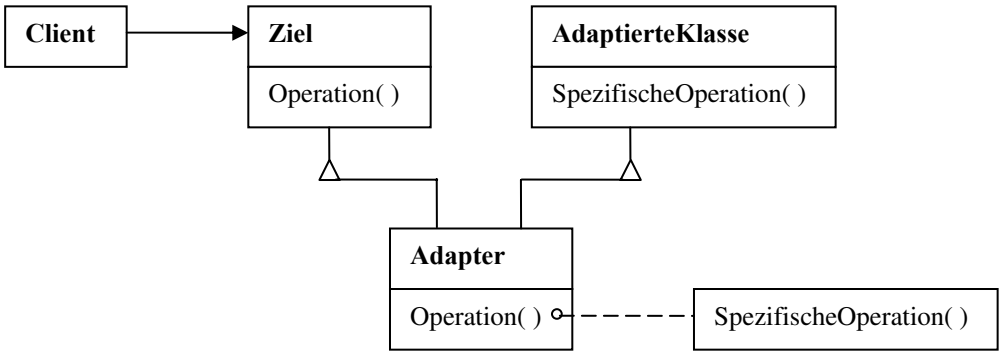


Abbildung 9: Struktur eines Adapters

[108] Vgl. [Gamm 1996] S. 8 ff
[109] Vgl. [Gamm 1996] S. 171 ff

Dabei greift der Client auf die Zielklasse zu und möchte eine Operation ausführen, die die Klasse nicht bietet, aber von der adaptierten Klasse angeboten wird. Auf direkten Wege kann diese Operation aus z.B. Inkompatibilitätsgründen nicht erfolgen, deswegen wird in diesem Entwicklungsmuster eine neue Klasse „Adapter“ eingeführt, die beide Methoden von den Oberklassen vererbt bekommt, die Berechnung ausführt und das Ergebnis an den Client zurückgibt.

Vor- und Nachteile von Entwurfsmustern seien hier zusammengefasst:

Bewertung von Entwurfsmustern[110]	
Vorteile	**Nachteile**
• Fördern das Verständnis und die Identifizierung von Objekten, auch solchen die nicht in der Realität vorkommen, aber für die zu erstellende Software nötig sind • Praxisorientierung und Hilfestellung durch umfangreiche Informationen • Wissenstransport erfahrener Programmierexperten • Vereinfachte Kommunikation • Erleichterung der Sichtweise auf komponentenbasierte Software-Entwicklung und Wiederverwendung • Gute Wartbarkeit durch geringe Anzahl der Klassen • Flexible Erweiterbarkeit und Kosteneinsparung • Sehr bekannt und häufige Anwendung durch hohe Akzeptanz • Höhere Abstraktion bei Entwurf, Dokumentation und Forschung • Vereinfachte Wartung	• Entwicklungsmuster können nicht alle Probleme abdecken • Einige Muster sind schwer verständlich und werden selten genutzt • Verhindern selbständiges kreatives Problemlösen • Einfachere Lösungen werden teilweise nicht mehr angestrebt, Effizienzeinbußen sind die Folge • Hoher Lern- und Suchaufwand • Sehr hoher Entwicklungsaufwand neuer Muster • Keine Automatisierbarkeit

Tabelle 18: Bewertung von Entwurfsmustern

Neben dem Entwurfsmuster gibt es unter anderem **Architekturmuster**, die prinzipiell das gleiche Ziel verfolgen, nämlich ein spezifisches Problem durch eine bestehende Schablone zu

[110] Vgl. [Gamm 1996] S. 4 ff und [Vran 2002] S. 1 ff

lösen. Architekturmuster sind hierbei wesentlich weitgreifender, da sie ein komplettes Grundgerüst für die Anordnung von Elementen in einem Softwaresystem vorgeben.

3.2.4.2 Polymorphismus

In den objektorientierten Sprachen gibt es Möglichkeiten, Teile eines Programms einer ordnenden Betrachtung zu unterziehen und so Komponenten abzugrenzen.[111] Dies kann z.B. das zentrale Konzept des Polymorphismus, die Vielgestaltigkeit von Methoden in der Objektorientierung sein. Er ist „die Möglichkeit, Objekte passender Schnittstellen zur Laufzeit füreinander einzusetzen.“[112]

So gibt es unterschiedliche Arten des Polymophismus:[113]

- **ad hoc Polymorphismus**:
  (= unterschiedlicher Code wird für gleiche Objekte verwendet)
  - Überladen von Werten
  - Erzwungener Polymorphismus (Coercion) - Veränderungen von Datentypen bei Operationen mit unterschiedlichen Datentypen
- **universellen Polymorphismus:**
  (= gleicher Code wird für unterschiedliche Objekte verwendet)
  - Parametrischer Polymorphismus - Erstellung von Templates
  - Inklusionspolymorphismus - Einschließung einer Komponente durch eine andere

Gleichnamige Methoden können so auf Objekte unterschiedlicher Klassen angewendet werden. Methoden der Superklassen werden bei Nutzung einer gleichnamigen Methode in Subklassen ersetzt, man spricht von **Überladung**. Wichtig ist hierbei nur die passende Schnittstelle des Objektes, andere Informationen z.B. über die Klasse, die dem Objekt angehört, sind nicht relevant. Damit ist man beim Stellen einer Anfrage vor der Ausführung nicht auf eine bestimmte Implementierung festgelegt. Gerade bei generischen Komponenten wie Templates kommt der Polymorphismus zum Tragen. Entscheidend ist lediglich die Kompatibilität. Objekte können auch während der Laufzeit ausgetauscht werden, so dass Objekte unterschiedlicher Klassen bei gleichen Anfragen unterschiedlich reagieren. Die Festlegung, welche Methode der Sub- oder Superklasse auf das Objekt angewendet wird, erfolgt erst zur Ausführungszeit. Dies wird das **Prinzip der späten Bindung** genannt[114].

---

[111] Vgl. [Baue 2000] S. 53 ff
[112] Vgl. [Gamm 1996] S. 447 ff
[113] Vgl. [Baue 2000] S. 53
[114] Vgl. [Balz 2001] S. 851

Die bereits vorgestellten Entwurfsmuster nutzen ebenso den Polymorphismus. Dadurch helfen sie, Schnittstellen und deren Elemente und Daten zu identifizieren und damit auch flexibel wiederverwendbare Software herzustellen.[115]

Bewertung des Polymorphismus[116]	
Vorteile	**Nachteile**
• Erspart aufwändige Switchanweisungen • Festlegung von anzuwendenden Methoden erfolgt erst bei der Ausführung des Codes • Mehrmalige Verwendung von Namen möglich • Der Sender muss lediglich wissen, dass der Adressat die Nachricht versteht • Rechtevergaben müssen bei Modifizierung der Objekte nicht überprüft werden • Effizienzverbesserungen bezüglich Zeit und Speicherplatz bei großen Programmen • Reduziert den Code-Umfang und vereinfacht das Handling • Fördert die Entwicklung wiederverwendbarer Komponenten • Das Prinzip des Polymorphismus kann selbst in anwendungsneutralen, wiederverwendbaren Klassenkonstrukten entwickelt werden • Polymorphismus ist auf jede Methode anwendbar	• Implementierung sehr aufwändig und bedarf Experten • Testbarkeit schwierig umzusetzen • Programmablauf nicht aus dem Quellcode ableitbar • Alle dynamischen Abläufe müssen getestet werden • Wird Polymorphismus mehrfach wiederholt, explodiert die mögliche Anzahl der Ablaufpfade

Tabelle 19: Bewertung des Polymorphismus

[115] Vgl. [Gamm 1996] S. 17

[116] Vgl. [Balz 2001] S. 828 ff, [Gamm 1996] S. 19 ff, [Baue 2000] S. 53 ff und [Gruh 2004] Folie 1 ff

3.2.5 Komponentenarchitekturen

Komponentenarchitekturen sind das Herzstück der Systemfamilienentwicklung, weil sie die „Rohstoffe“ systematisch zusammenbauen. Die Zusammenhänge seien hier grafisch dargestellt:

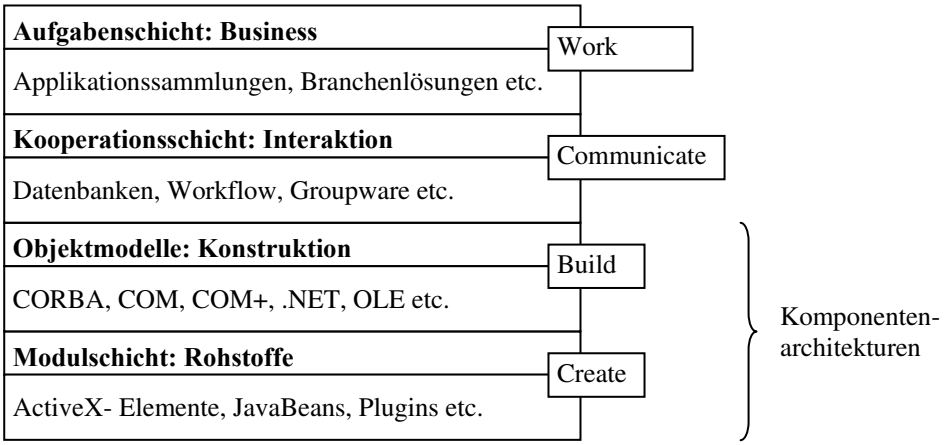


Abbildung 10: Schichtenmodell der komponentenbasierten Software[117]

Komponentenarchitekturen seien im Folgenden vorgestellt. Unterschieden wird dabei zwischen Einzelplatzsystemen wie COM, und Architekturen für verteilte, heterogene Systeme wie z.B. CORBA.

3.2.5.1 COM / ActiveX von Microsoft

COM ist das objektorientierte Komponentenmodell von Microsoft. Der Ursprung von COM liegt in der in den 1990ern eingeführten Technik der Verbunddokumente OLE, bei denen Teildokumente in andere Dokumente eingebettet werden. Grundidee ist hierbei, dass Anwendungen Funktionen bereitstellen, die auch anderen Programmen von Nutzen sein können. So muss z.B. Word wissen, wie es eine integrierte Exceltabelle darstellen muss. Dies bedeutet eine Wiederverwendung von bereits bestehendem Programmcode und damit Wiederverwendung von Komponenten.

Der Datenaustausch erfolgt im Client/Server-Modell: Der COM Server erzeugt und nutzt Objekte und bietet die zu exportierenden Klassen, die COM-Objekte, über die Schnittstellen an. In den Schnittstellen selbst ist die Information hinterlegt, welche Funktionen angeboten werden. Die Nutzung der Funktionen erfolgt mit Hilfe von Zeigern auf eine oder mehrere Schnittstellen einer Komponente. Die Schnittstelle selbst besitzt einen Zeiger auf ihre interne Tabelle, welche die Zeiger auf ihre Funktionen verwaltet. Es kann nie direkt auf Objekte zugegrif-

[117] Vgl. [Weis 1999] S. 21

fen werden. Vererbung ist hier jedoch nur bei den Eigenschaften der Schnittstellen möglich. Schnittstellen haben dabei eine weltweit eindeutige GUID, die vor der Nutzung der Komponente in die Windows Registry eingetragen werden muss. Durch die eindeutige Identifizierung über die GUID können Komponenten auch den gleichen Namen tragen.[118]

Komponenten liegen hier als Binärstandard vor und sind damit programmiersprachenunabhängig, die Spezifikation der Schnittstellen enthält keine spezielle Syntax, sondern beschreibt den Aufbau und die angebotenen Dienste von Komponenten. Daneben wurde 1996 ein weiteres Modell, das DCOM für verteilte Anwendungen im Netzwerk geschaffen. Heute entwickelt Microsoft COM+[119] als eine Erweiterung und die .NET-Frameworks[120], die einige Probleme von COM verbessern sollen und laut Microsoft COM ablösen werden.

COM-Komponenten werden als Module in Form von Anwendungen (als EXE auf dem lokalen Server mit langsamerer Ausführung), als DLLs und OCX (prozessintern und damit schnell laufend) oder als Windows Script Components ausgeliefert.[121]

Seit 1996 gibt es auch COM-Komponenten, die speziell für das Internet geschaffen worden sind: ActiveX Steuerelemente. Sie haben nur eine Schnittstelle und die Fähigkeit, sich automatisch in die Windows Registry einzutragen. Solche Steuerelemente werden dabei fest auf dem Client installiert, was massive Sicherheitsprobleme mit sich bringt, da so das zugrunde liegende Betriebssystem offen liegt.

[118] Vgl. [Balz 2001] S. 871 ff

[119] COM+ ist eine in Binärform vorliegende Plattform und ein Modell für die Clientseite in einem, und gehört propritär zu Windows. Vgl. [Balz] S. 941

[120] .NET ist ein microsoftbasiertes Bundle von Softwaretechnologien zur Softwareentwicklung, bestehend aus einer virtuellen Laufzeitumgebung und einem Framework von Klassenbibliotheken und Diensten.

[121] Vgl. [Schw 2005]

Die Architektur von COM kann wie folgt bewertet werden:

Bewertung der COM-Architektur[122]	
Vorteile	**Nachteile**
• Programmiersprachenunabhängige Schnittstellen • Wiederverwendung von bestehendem Programmcode • Flexibles und mächtiges Komponentenmodell • Kein direkter Zugriff auf Objekte • Namenskollisionen durch die GUID ausgeschlossen • Erweiterbare Zusatzfunktionen durch einfache Integration von Komponenten • Unterstützung • Schnelle Ausführung • Hohe Marktdurchdringung und Entwicklungspotential • Ermöglicht mehrschichtige Architekturen und verteilte Anwendung mittels des Microsoft Transaction Server, kurz MTS • Portabler Einsatz • Dynamische Bindung möglich • Mehrere Interfaces	• Propritär für Windows entwickelt und damit auch kostenintensiv • COM selbst ist als Einzelplatzsystem gedacht, Abhilfe schafft hier COM+ und das neue .NET, welches aber einen hohen Anfangsaufwand braucht (Verständnis, Integrationszeit) • ActiveX Steuerelemente können durch Offenlegung des Betriebssystems Sicherheitslücken verursachen • Versionierung wird nicht beherrscht • Sehr kompliziert zu verwenden • Ingnoriert jeden Standard • Verträgt sich schlecht mit Garbage Collectoren

Tabelle 20: Bewertung der COM-Architektur

3.2.5.2 JavaBeans von Sun

JavaBeans sind ein Komponentenmodell basierend auf der Programmiersprache Java. Ihr Ziel ist es, die Zusammenstellung von Komponenten für ein komplettes Programm mit minimalem Aufwand über grafische Oberflächen zu ermöglichen. Auch einzelne Beans lassen sich als Applets z.B. auf Webseiten einbauen. In Builder Tools können JavaBeans per Mausklick zusammengefügt werden. Prinzipiell ist jede Bean eine Java-Klasse, die auf Konfiguration und Anpassbarkeit modifiziert wurde. JavaBeans legen für die nachträgliche Anpassbarkeit ihre Eigenschaften, Methoden und Ereignisse in einem Builder offen und sind somit introspektiv. In ihnen sind indizierte (Eigenschaften können mehrere Werte annehmen), gebundene (teilen

[122] Vgl. [Schw 2005], [Balz 2001] S. 896, [Micr 2005], [Weis 1999] S. 19 ff und [Punz 1999]

anderen Beans Änderungen mit) und Eigenschaften mit Nebenbedingungen (andere Beans können vor einer Änderung ein Veto einlegen) implementiert. Verbunden werden sie lose durch Ereignisse mittels get( ) und set( ) Methoden, eine JavaBean kann so ihre Dienste anderen Javabeans anbieten und wiederum andere Dienste beanspruchen, indem sie sich als Empfänger bzw. Abhörer bei der entsprechenden Bean anmeldet. Diese lose Kopplung macht JavaBeans sehr flexibel einsetzbar.[123] JavaBeans benötigen dabei immer einen Container, also eine Laufzeitumgebung, in dem sie eingebettet werden können.[124] Für die Entwicklung von JavaBeans hat Sun Spezifikationen herausgegeben, um die Einhaltung des Standards sicher zu stellen.

Bewertung der JavaBeans[125]	
Vorteile	**Nachteile**
• Architektur- und Plattformunabhängigkeit „Write Once Run Anywhere“ • Nicht typgebunden • Strukturierte Anwendungsentwicklung • Einfache Erstellung, da Java • Starke Unterstützung von Seiten der Industrie (Apple, Borland, IBM, JustSystem, Microsoft, Netscape, Rogue Wave, SunSoft and Symantec etc.) • Großes Anwendungsspektrum (komplexe traditionelle Anwendungen bis hin zur Integration einzelner Beans) • Einfache und vor allem schnelle Erweiterung bestehender Programme • Portierbarkeit • Schnelle Anpassbarkeit • Nutzung der vielen Java-Sicherheitsmaßnahmen • Spätes Binden als Teil der Java Sprache verwirklicht • Graphische Verbindung von Komponenten bereits im Sprachumfang de facto normiert.	• Langsame Ausführung, da sie auf der Java Virtual Machine erst interpretiert werden müssen • Schwieriger Zugriff auf Datensystem und Hardware durch Sicherheitsschranken • Keine Brücke von ActiveX in Richtung JavaBeans • Keine Ausnutzung der Java Interfaces • Keine strenge Trennung von Interface und Implementation

Tabelle 21: Bewertung der JavaBeans

[123] Vgl. [Balz 2001] S 856 ff
[124] Vgl. [Java 2005]
[125] Vgl. [Balz 2001] S. 896 und [Java 2005] und [Weis 1999] S. 19 ff und [Lemp 2002] S. 9 ff und [Punz 1999]

3.2.5.3 CORBA von OMG

CORBA gehört zu den verteilten objektorientierten Anwendungen und unterstützt als Middleware genau wie COM+ oder wie die EnterpriseJavaBeans Geschäftsabläufe in verteilten und heterogenen Systemen. Middleware-Plattformen müssen also Mehrbenutzerfähigkeiten besitzen, auf die Anzahl der Nutzer, Daten und Geschäftsprozesse skalierbar sein und eine hohe Verfügbarkeit besitzen.

CORBA ist dabei lediglich ein Standard und eine Spezifikation für die Infrastruktur solcher Plattformen und Komponentenmodelle, die ausschließlich über Schnittstellen kommunizieren. Die Basis dafür ist die Referenzarchitektur Object Management Architecture, kurz OMA:

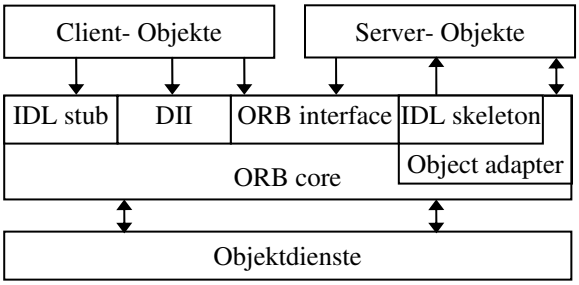


Abbildung 11: OMA Architektur von CORBA[126]

Komponenten dieser Architektur sind die Client- und Serverobjekte und der Object Request Broker, kurz ORB. Dieser leitet die Anfrage zwischen den Client- und Serverschnittstellen im verteilten System weiter, um nicht auf komplizierte Netzwerkprotokolle zurückgreifen zu müssen. Das ganze geschieht mit Hilfe von Stubs auf der Clientseite und Skeletons auf der Serverseite. Sie simulieren die angebotenen Objektdienste wie Namensgebung, Kopieren und Löschen von Objekten (Lifecycle-Dienste), Hilfsdienste, Drucken, Sicherheitsdienste, als wären sie lokal vorhanden. Die Kommunikation erfolgt ausschließlich darüber. Auf die Objekte selbst kann nie direkt zugegriffen werden. Schnittstellen, Klassen und Objekte werden dabei mit Hilfe der Definitionssprache IDL beschrieben. Die Schnittstellenbeschreibung selbst liefert das Dynamic Invocation Interface (DII) und ersetzt unbekannte Schnittstellen.

Der erläuterte Aufbau ist äquivalent zu den EnterpriseJavaBeans.[127]

[126] Vgl. [Balz 2001] S. 925 und 927

[127] Vgl. [Balz 2001] S. 924 ff und [Wiki 2005] CORBA

Zusammenfassung der Vor- und Nachteile des Middleware-Standards:

Bewertung von CORBA[128]	
Vorteile	**Nachteile**
• Offener Standard mit uneingeschränkter Erweiterbarkeit • Programmiersprachen und Plattformunabhängigkeit • Verbesserte Internetanwendungen • Beliebige Einbindung von CORBA-Objekten • Für mobile Geräte gibt es eine schlankere CORBA-Form • Ausfallsichere Middleware mit einfacher Kommunikation • Skalierbarkeit • Spezifikation zur Entwicklung verteilter Anwendungen / Komponenten • Starke Unterstützung seitens der Industrie • Portables System • Komplementäres Arbeiten mit EJB • Strenge Trennung von Interface und Implementierung • Dynamische Bindung	• Rechenintensive Architektur • Vererbung kann möglich sein: Der CORBA Standard gibt keine Hinweise wie Stubs und Skeletons zu funktionieren haben. Deshalb kann durchaus Vererbung intern verwendet werden mit allen Nachteilen. • Keine Möglichkeit des Multiple Interface Handling: Damit gibt es auch keine Möglichkeit des geregelten Updates einzelner Komponenten von CORBA Systemen.

Tabelle 22: Bewertung von CORBA

3.3 Konfiguration im laufenden Betriebssystem

3.3.1 Windows

3.3.1.1 Beschreibung der Konzepte

Microsoft hat für die Software für Privatnutzer im Lauf der Zeit unterschiedliche Konzepte ent- und weiterentwickelt.

Unter **MS-DOS** hatten alle Programme eigene Konfigurationsdateien für anwendungsbezogene Daten. Das Betriebssystem speicherte eigene Einstellungen in 2 Dateien: Der config.sys und autoexec.bat. **Windows 3.0** führte eine systemweite Einstellungsspeicherung in 4 Initiali-

[128] Vgl. [Weis 1999] S. 19 ff und [Balz 2001] S. 924 ff und [Punz 1999]

sierungsdateien ein: Program.ini, control.ini, win.ini und system.ini, einfache lineare und unverschlüsselte Anweisungen in Textdateien. Die Verwendung der INI-Dateien war und ist relativ einfach zu bewerkstelligen und bietet einen geringen Aufwand für die Software-Programmierer. Schnittstellen sind nicht vorgesehen und beschränken die Funktionalität von Software, da nur im Rahmen der systemweiten INI-Dateien von Windows auf andere Programme oder Ressourcen zugegriffen werden kann.

Dieses System hat aber einige Nachteile:

Nachteile der INI-Dateien / Textbasierender Konfigurationen[129]
• Da sie textorientiert sind, beträgt ihre Größe maximal 64 KB[130], das begrenzt die Anzahl möglicher installierter Programme • Zugriffe auf große lineare Dateien sind im Gegensatz zu einer Datenbank langsam • Redundanz der Anweisungen • Gelöschte Programme hinterlassen Einträge, auch Programme mit eigenen INI-Dateien hinterlassen „Datenleichen“ • Alle Einstellungen werden global gespeichert, Systempflege ist nicht möglich • Netzwerke werden aufgrund fehlender APIs nicht unterstützt

Tabelle 23: Bewertung der INI-Dateien / Textbasierender Konfigurationen

Deswegen wurde mit **Windows 3.1** ein erster zentraler Datenspeicher für betriebssystemweite Konfigurationen, die reg.dat, eingeführt. Dieser lag in einem undokumentierten Binärformat vor und konnte nur per Registry-Editor geändert werden. Die zugehörige Datei durfte aber nicht größer als 64 KB[131] werden, unterstützte noch nicht die automatische Synchronisation der Daten und persönliche Einstellungen waren nicht portabel.[132]

Seit **Windows 95** wurde die Registry[133] verbessert. Konfigurationen des Systems werden in logisch zusammenhängender Form in einer hierarchischen zentralen Datenbank gespeichert und verwaltet. Die Daten werden über Schlüssel aufgerufen und sind untereinander verlinkt. Die Informationen werden dabei in zwei verschlüsselten Dateien im Windowsverzeichnis gespeichert: Die user.dat und die system.dat[134]. Zusätzlich zur Registry werden aus Kompatibilitätsgründen noch die alten INI-Dateien win.ini und system.ini der Win 3.x Versionen ge-

[129] Vgl. [Robi 1998] S. 1 ff

[130] Vgl. [Robi 1998] S. 4

[131] Diese Grenze wurde aber mit Windows NT 3.1 aufgehoben. vgl. [Robi 1998] S. 6

[132] Vgl. [Robi 1998] S. 5

[133] Auch Registrierung oder Registrierungsdatenbank genannt.

[134] Die Nutzerdatei user.dat enthält spezifische Daten des angemeldeten Nutzers, die im Netzwerk überall verfügbar sein müssen, die Systemdatei system.dat enthält die lokale Systemkonfiguration für den Start des Betriebssystems und die Anwendungsausführung, also Hardware, Bibliotheken und Dateien.

pflegt, da alte 16-bit Anwendungen noch auf sie angewiesen sind. Die Registry lässt mehrere Konfigurationen verschiedener Nutzer oder wechselnde Systemkonfigurationen zu und enthält alle Hardware- und Betriebssystemparameter. Zusätzlich ermöglichen Sicherungsmaßnahmen eine Wiederherstellung der Registry nach einem Systemabsturz. Netzwerkfunktionen einschließlich Remote-Access sind möglich.[135] Die Registry speichert die Informationen in einem hierarchisch aufgebauten, untereinander verlinkten und damit redundanzfreien und einheitlich aktualisierbaren Schlüsselsystem. Die enthaltenen Werte können Subschlüsseln zugeordnet werden. So werden Einstellungen in Gruppen zusammengefasst. Der Baum besteht je nach Betriebssystem aus 5 oder 6 Hauptschlüsseln, jeder enthält einen bestimmten Aspekt der Systemkonfiguration; Benutzer- und Maschinendaten werden getrennt:

Aufbau der Registry von Windows XP	
HKEY_CLASSES_ROOT	OLE, Drag & Drop, Befehle und Icons von Datentypen, Schnellansicht (Quick View), verschiedene Routinen für Programme
HKEY_CURRENT_USER	Daten über den aktuell im System angemeldeten Nutzer, Einstellungen und Pfade von Sound, Bildschirmangaben, Installationspfade, Netzwerkeinstellungen, Remote-Zugriff, Einstellungen von Software, letzte aufgerufene Dateien
HKEY_LOCAL_MACHINE	Treiber, installierte Hardware, Schnittstellen und Ports, Software-Einstellungen, Netzwerk und Sicherheit, installierte Software, Startdaten
HKEY_USERS	benutzerspezifische Informationen, Standardeinstellungen für neue Nutzer, Subkeys für jeden Nutzer
HKEY_CURRENT_CONFIG	Plug and Play Daten, Informationen über die aktuelle Hardware-Konfiguration
HKEY_DYN_DATA	Schlüssel für dynamische Daten von Diensten, Hardwarekonflikte und Gerätestatus

Tabelle 24: Aufbau der Registry unter Windows XP

Die Registry ist codiert, von manuellen Änderungen der hochsensiblen Daten wird in jeder Literatur abgeraten. Vorherige Sicherheitskopien sind eine obligatorische Empfehlung. Änderungen der codierten Schlüssel[136] und Werte können das Betriebssystem unbrauchbar machen. Für die Modifikation der Parameter gibt es verschiedene Möglichkeiten.

[135] Vgl. [Born 1998] S. 4 ff

[136] Die Schlüssel in der Registry setzen sich aus diversen Daten wie der MAC-Adresse und der Uhrzeit zusammen und dienen der Identifikation von Schnittstellen zu DLLs und Windows Komponenten (COM- Komponenten), deswegen sollten sie auf keinen Fall verändert werden.

Der wichtigste Weg ist der Registrierungseditor:

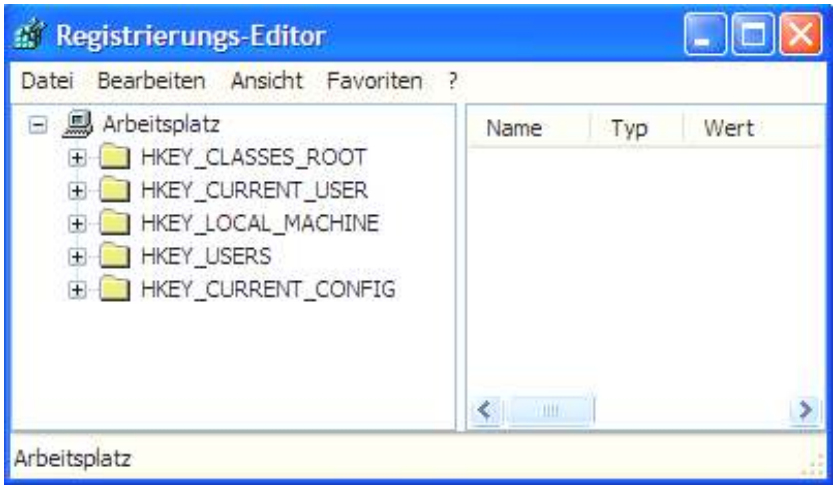


Abbildung 12: Der Registrierungs-Editor

Zudem gibt es auf dem Markt von Drittanbietern und von Microsoft selbst eine Vielzahl von Tools zur komfortablen Optimierung und „Tuning“ der Registry, zum Aufräumen von Fehlern und Inkonsistenzen sowie zum leichten Durchsuchen nach Einträgen[137]. Microsoft unterstützt die Konfigurierbarkeit durch die Systemsteuerung und verschiedenen Reitern wie z.B. den Bildschirmeigenschaften; Änderungen werden auch durch Setup- und Installationsprogramme über eingerichtete APIs in der Registry vorgenommen. Programme selbst bringen diverse Konfigurationsmenüs für Maus, Drucker, Tastatur etc. mit sich.

Viele moderne Programme verwenden nach wie vor INI-Dateien für die Hinterlegung der Konfiguration und verzichten auf Schnittstellen zu den Registrydaten. Aus sicherheitstechnischen Gründen werden Serial-Nummern aber üblicherweise in die Registry, und damit verschlüsselt, eingetragen.

Die Verwendung der beiden Systeme kann aber zu Konflikten führen, denn bei jedem Programmstart werden INI-Dateien mit Registryeinträgen verglichen und unter Umständen überschrieben.[138] Ein weiterer Vorteil des Registry-Konzeptes ist der Im- und Export von zusammengehörigen Daten. So können bestimmte Schlüssel und somit Konfigurationen wie z.B. die Datentypen im Netzwerk ausgetauscht werden. Diese Methode wird auch von Softwareherstellern und Systementwicklern verwendet, die so einfach über Reg-Dateien Informationen in die Registry importieren.[139]

[137] Vgl. [Born 1998] S. 51 ff
[138] Vgl. [Born 1998] S. 12 ff
[139] Vgl. [Born 1998] S. 26 ff

3.3.1.2 Bewertung der Konzepte

Die Registry entstand aus den Bemühungen heraus, die Nachteile der einzelnen Systemdateien zu beseitigen. Das Konzept besteht aus der Trennung der PC-eigenen Daten und der Nutzerdaten, da die Anwendung auch im Netzwerk stattfinden soll. Die Konfigurationseinstellungen werden dabei verschlüsselt im Datenbankformat hinterlegt, die Daten sind global auf dem PC für alle Programme abrufbar, einheitlich und aktuell, Inkonsistenzen durch veraltete Bibliotheken oder Datentypendefinitionen sind nicht möglich. Durch permanente Prüfungen und das Sicherungskonzept lassen sich selbständige Reparaturen und Restaurierungen durchführen. So werden „Datenleichen“ bei z.B. der Deinstallation eines Programms eliminiert.

Problematisch ist, dass alle Daten voneinander abhängig gespeichert sind. Das macht das System fehleranfällig. Zudem werden die Schnittstellen von Programmierern nicht zwangsweise eingehalten. Das Ergebnis sind eigene redundante Konfigurationsdateien. Aus Kompatibilitätsgründen müssen auch die herkömmlichen INI-Dateien beibehalten werden. Durch die Verteilung im Netzwerk werden Daten aber auch sicherheitsrelevant sensibler.

Die Vorteile für ein großes Betriebssystem liegen jedoch auf der Hand, da hier zentral die wichtigen Einstellungen des Betriebssystems verwaltet werden; das Konzept selbst ist jedoch noch in einigen Punkten verbesserungswürdig.

Die Vor- und Nachteile des Datenbankkonzeptes seien hier im Folgenden noch einmal zusammengefasst:

Zusammenfassung des Datenbank-Konzeptes[140]	
Vorteile	**Nachteile**
• Speicherung in logisch zusammenhängender Form in einer zentralen Datenbank • Schneller Zugriff • Verlinkung garantiert konsistente Verwendung der gleichen Dateien auf aktuellstem Stand • Die Aufteilung auf 2 Dateien ermöglicht Netzwerkeinstellungen und dort global hinterlegte Nutzerprofile und trotzdem lokale Einstellungen • Remote-Zugriff • Im- und Export von Daten möglich • Austausch von Konfigurationen • Einheitliche Aktualisierung von Daten • Verringerung von Redundanzen • Starke Unterstützung durch Werkzeuge • Anzahl installierbarer Programme • Wiederfindung der Einstellungen • Mehrbenutzerbetrieb • wechselnde Systemkonfigurationen möglich • Sicherungssystem	• Das Modell selbst ist fehlerbehaftet, da die eigenen Ideen nicht immer eingehalten werden • Softwarehersteller halten sich nicht an die Schnittstellen • Konflikte bei der Überprüfung können zum Überschreiben der Daten führen • Verschlüsselung macht manuelle Änderungen schwierig und störanfällig • Hohe Fehleranfälligkeit • APIs werden nicht eingehalten • Das System ist selbst nicht konsistent, wenn aus Kompatibilitätsgründen das alte Format beibehalten werden muss

Tabelle 25: Bewertung des Datenbank-Konzeptes

[140] Vgl. [Born 1998] S. 12 ff und [Robi 1998] S. 8 ff

3.3.2 Linux

3.3.2.1 Beschreibung der Konzepte

Unter Linux gibt es keinen allgemeingültigen Weg wie er bei Microsoft Windows üblich ist, da jede Entwicklergruppe eigene Ansichten und Ziele verfolgt. So ist jede Distribution anders aufgebaut. Grundlage dieser Entwicklungen ist die Tatsache, dass der Linux Quellcode für das freie Betriebssystem unter der GNU General Public License freigegeben ist, jeder darf es verwenden, kopieren, weitergeben und verändern[141]. Hier wird neben dem Kernel und den Softwarepackages auch die grafische Oberfläche aus der Vielzahl der möglichen Varianten[142] installiert; die einzelnen Bestandteile des Betriebssystems sind nicht wie bei Microsoft in ein Komplettsystem eingebettet sondern modular verfügbar, und erklären so die Komplexität von Linux[143]. Bei der Oberfläche wird häufig GNOME, KDE, Fluxbox / Blackbox, Xfce 4 oder eben nur die Konsole verwendet. Alle verfolgen ein anderes Konfigurationsschema.

Linux ist damit aber auch transparenter aufgebaut und in allen Bereichen vom Nutzer konfigurierbar. Somit kann es von Experten leichter bedient und an die eigenen Ansprüche angepasst werden. Vorkompilierte Datenpakete (RPMs) gibt es deswegen im Internet auf diversen Seiten zum kostenlosen Download.[144] Das System hat Vorteile gegenüber Microsoft Windows: Es kann besser angepasst werden und Sicherheitslücken werden schneller entdeckt und beseitigt. Der entscheidende Nachteil ist, dass Linux einer komplexeren Installation bedarf; es sei denn man greift zu gängigen Distributionen wie z.B. Red Hat oder Debian, Suse Linux, oder Mandrake[145]. Die verschiedenen Distributionen verwenden dabei eigene Tools die bei der Konfiguration helfen, inzwischen gibt es aber auch systemübergreifende Werkzeuge wie z.B. Webmin[146]. Einerseits sind sie in jedem Linux verwendbar, hinken aber eben der Entwicklung hinterher, da sie jeder Anforderung gerecht werden wollen.

Für diese Arbeit wird die Distribution Suse Linux 9.2 verwendet. Um Konfigurationen an diesem System vorzunehmen, werden der Systemsteuerung ähnliche Oberflächen von KDE 3.3 und YaST 2, die Kontrollzentren, verwendet.

---

[141] Vgl. [Wiki 2005] Linux

[142] Eine sehr gute Übersicht beliebter Linuxdistributionen gibt es unter [Fres 2005]

[143] Vgl. [For-D 2005] Übersicht

[144] [Saou 2005] oder auch [Rpms 2005] bieten Software als fertig kompilierte und angepasste Pakete an.

[145] Vgl. [For-D 2005] Übersicht

[146] Vgl. [Lin-W 2005] Webmin

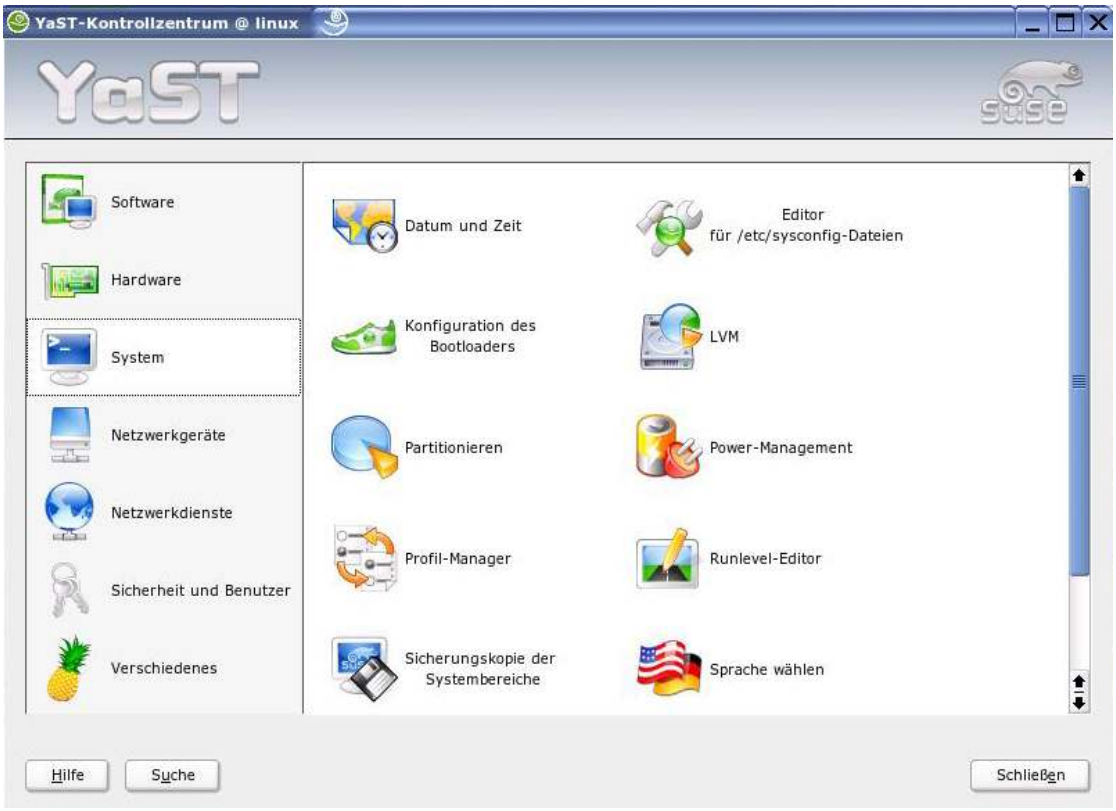


Abbildung 13: YaST 2 - die Systemsteuerung für Suse Linux

Konfigurieren lässt sich Linux im Allgemeinen nach dem Leitsatz: „Linux is made with one thought in mind: Everything is a file.“[147]. Dabei gibt es gravierende Unterschiede zu Microsoft Windows wie z.B. die Dateisystemstruktur, die Benutzerrechte und der Betriebssystemaufbau. Aufgrund der Vielzahl möglicher Konfigurationen in den verschiedenen Packages gibt es viele Wege, die Einstellungen an einem Linuxsystem zu modifizieren. Auch andere Anbieter folgen dem Beispiel von Suse und bieten benutzerfreundliche Oberflächen für das Einstellungsmanagement an.

Alle Änderungen, die über diese Oberflächen vorgenommen werden und systemrelevant sind, speichert Suse im Ordner etc/ im Verzeichnis sysconfig und überträgt die Konfigurationsdaten in die entsprechenden einheitlichen Systemdateien. Die Daten werden dabei genauso formatiert wie die Windows INI-Dateien: Schlüsselnamen für die Konfigurationsgruppe werden von unverschlüsselten Anweisungen ergänzt.[148] Als Beispiel sei hier die Konfigurationsdatei der VDR-Software für die Sendekanäle dargestellt:

[147] Vgl. [Xian 2005] Beginner
[148] Vgl. [Kofl 2003] S. 561

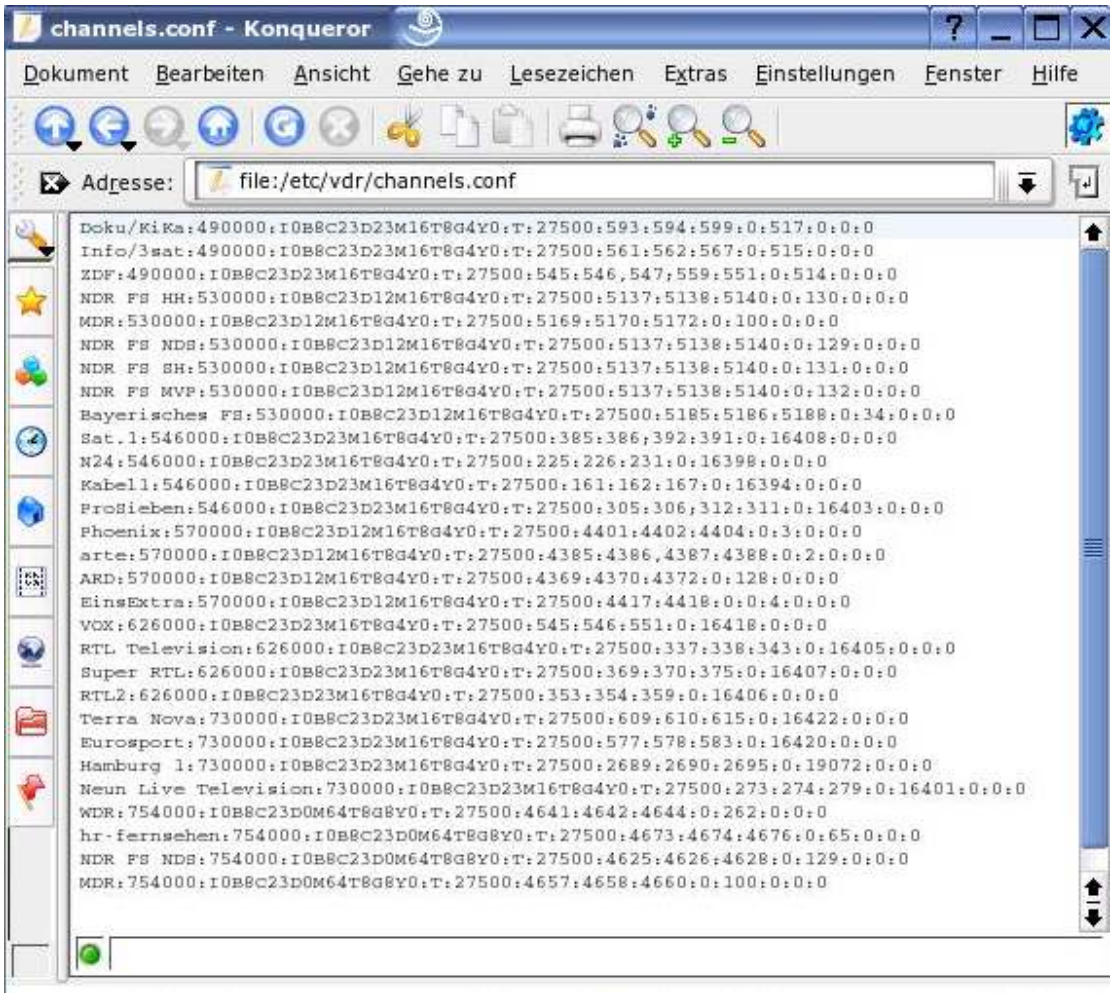


Abbildung 14: Konfiguration der VDR-Kanäle

Hinter der Konfiguration der Einstellungen unter Linux stecken aufgrund der unterschiedlichen Entstehungsgeschichten gleich mehrere Möglichkeiten für das Konfigurationsmanagement von Anwendungen und dem Betriebssystem selbst. Mit der Zeit haben sich einige Konventionen entwickelt die von vielen Programmen benutzt werden. Ursprünglich werden reine Textdateien verwendet, die sich allerdings in der Syntax erheblich unterscheiden. Manche Anwendungen verwenden in den zugehörigen Konfigurationsdateien wie unter Windows einfache Zuweisungen der Form „Schlüsselname = Wert“, andere Anwendungen nutzen für die Speicherung der Einstellungen eine C-ähnliche Syntax[149]. Auch Mischungen zwischen verschiedenen Stilen gibt es, so verwendet z.B. Apache eine Art Markup-Sprache, die in ihrer Art einer Mischung aus HTML und herkömmlichen Wertzuweisungen ist.[150]

[149] Die Programmiersprache C ist eng mit der Entstehungsgeschichte von Unix verbunden, denn die Sprache wurde auf dem jungen Betriebssystem entwickelt. Unix selbst und alle Programme für Unix wurden anfänglich vollständig in C geschrieben.

[150] Vgl. [Apac 2005] FAQ

Andere Programme wie das hier verwendete Openoffice.org 1.1.4 verwenden das XML-Format. Dies hat den Vorteil, dass es von vielen leistungsfähigen Parsern unterstützt wird und eine gängige Grundstruktur für Weiterentwicklungen und Änderungen bietet und sich für Entwickler leichter lesen lässt. Die Desktop-Oberfläche GNOME schlägt ebenso einen zukunftsweisenden Weg ein: Zur Speicherung der Konfiguration des Systems wird eine Art von Registry-Datenbank verwendet.

Als Beispiele werden hier 3 verschiedene Strategien des Konfigurationsmanagements erläutet.

a. Das Registrierungsdatenbank-Konzept von GNOME 2.8

Für die grafische Oberfläche GNOME wurde ab der 2.0 Version eine Registry-ähnliche Datenbank eingeführt, um Programmeinstellungen, Anwendungs- und Benutzereinstellungen zentral zu verwalten. Die Datenbank GConf enthält aber gegenüber der Registry einige Verbesserungen: „So kann die Bibliothek mit variablen Backends zum Speichern der Daten arbeiten, was es neben der Verwendung des normalen XML-Formats ermöglicht, GConf-Daten in einer Berkley-Datenbank zu sichern. Weiterhin kann ein Administrator systemweite Vorgabewerte setzen -bei Bedarf sogar verbindliche. Zusammen mit einem LDAP-Backend wäre es sogar möglich, mehrere Computer zentral über ein Netzwerk zu administrieren. Ein weiterer Vorteil von Gconf ist, dass sich mehrere Programme, die auf dieselben Einstellungen zugreifen, nicht in die Quere kommen, und dass alle Einstellungsänderungen sofort an die betroffenen Programme gemeldet werden - sie treten ohne Verzögerung in Effekt. Damit die Einstellungen nicht so kryptisch wie in der Registry werden, sind sie außerdem in Schemas dokumentiert. Zum Editieren der GConf-Einstellungen enthält Garnome[151] die frühe Version eines grafischen GConf-Editors; ansonsten gibt es auch ein Kommando-Interface namens gconf-tool-2.“[152]

Der Zugriff auf die Datenbank erfolgt mittels des Konfigurationseditors, in dem Standard- und Vorgabewerte bearbeitet werden können und nach z.B. den zuletzt genutzten Schlüsseln, Werten und Beschreibungen gesucht werden kann.[153]

b. Das „Everything is a file“-Konzept von Suse Linux 9.2: YaST 2 und KDE 3.3

Für die Grundlage dieser Diplomarbeit wurde Suse Linux 9.2 installiert. Nutzern wird über YaST 2, Suses Installations- und Konfigurationsprogramm[154], das Konfigurieren erleichtert.

[151] Garnome ist das Entwicklungswerkzeug für den GNOME Desktop.

[152] Die Verbesserungen gegenüber der Windows Registry wurden [Lin-U 2003] entnommen.

[153] Mehr Informationen über die Registrierungsdatenbank von Gnome gibt es unter [GNO-D 2005] GNOME 2.8, [Lin-M 2005] Von der Aufzucht eines Zwerges, [Lin-M 2005] Bitte hier anmelden! und [Wiki 2005] GNOME

[154] Vgl. [Lin-W 2005] YaST

Die hier verwendete KDE 3.3-Oberfläche[155] bietet ebenso eine Vielzahl an grafischen Menüs, die die Einstellungen erleichtern und stark an die Systemsteuerung von Windows erinnern.

Beispielsweise kann die Optik der KDE-Kontrollleiste über einen Konfigurationsreiter geändert werden, die Änderungen werden vom Programm in den entsprechenden Verzeichnissen der Programme selbst oder im allgemeinen Verzeichnis /etc, häufig unter dem Dateikürzel conf hinterlegt. Ein schöner Aspekt, der für Linux und einfache Textdateien spricht: Änderungen, die über grafikbasierte Menüs erfolgen, können problemlos aufgefunden und nachvollzogen werden:

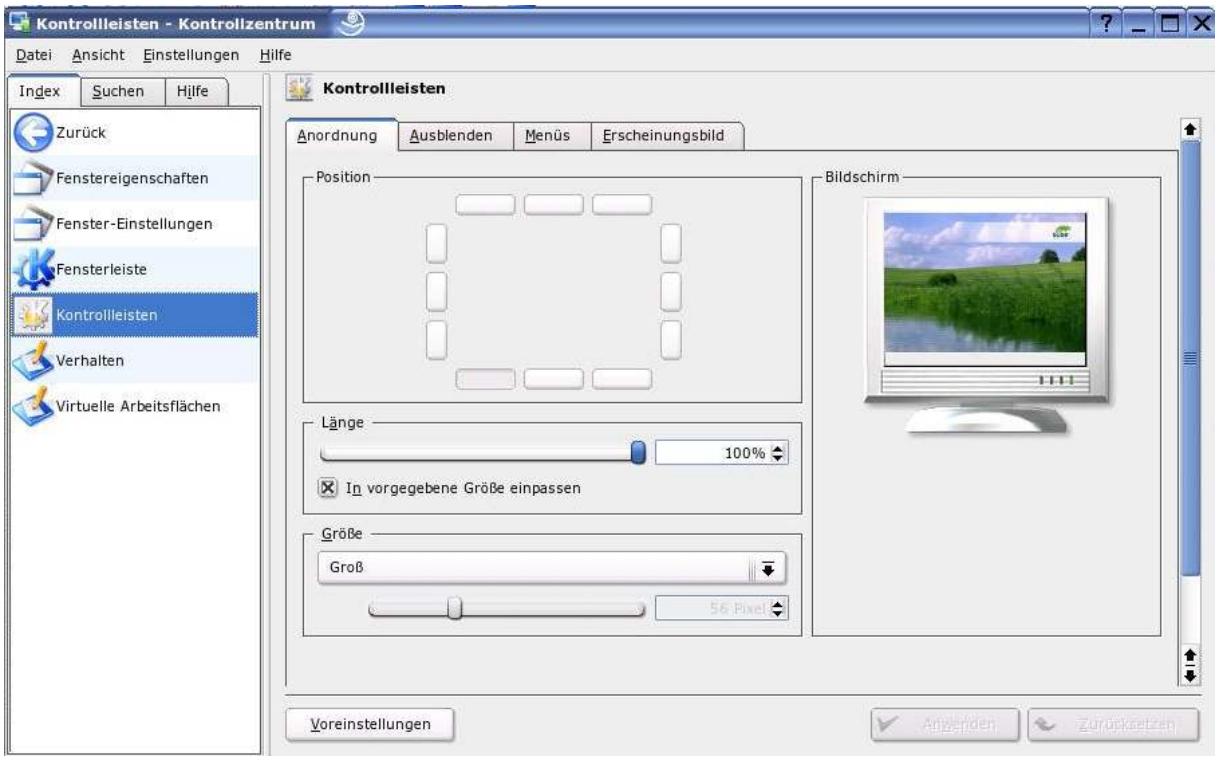


Abbildung 15: Ansicht des KDE-Kontrollzentrums unter Suse Linux 9.2

[155] Vgl. [Lin-W 2005] KDE

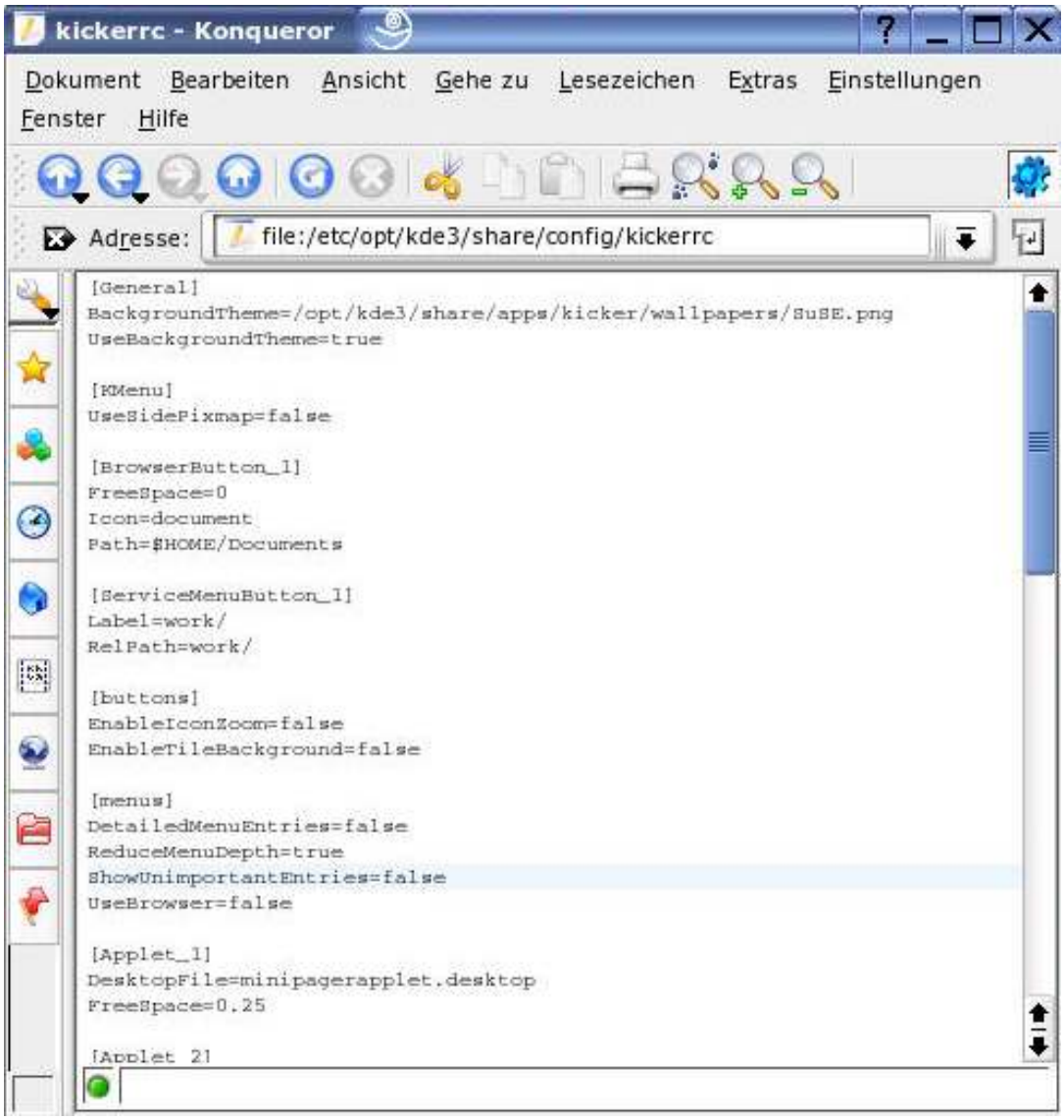


Abbildung 16: Die dazugehörige Konfigurationsdatei für die Kontrollleiste

c. Das XML-Konzept von Openoffice.org 1.1

Das freie Office-Paket Openoffice.org geht ebenso wie das Datenbankformat von GNOME einen modernen Weg, denn es speichert nicht nur seine Dokumente im XML-Format, sondern nutzt XML genauso zur Hinterlegung seiner Einstellungen.[156]

Man hat sich dabei für das XML-Format entschieden, da „XML ein Industrie-Standard ist, und somit die beste Wahl für anwendungsspezifische Dokumententypen darstellt“.[157] Beispielhaft sei hier ein Auszug einer Openoffice.org-Konfigurationsdatei für ein Userprofil dargestellt:

[156] Vgl. [OOo 2005] FAQ
[157] Vgl. [OOo 2005] Main FAQ

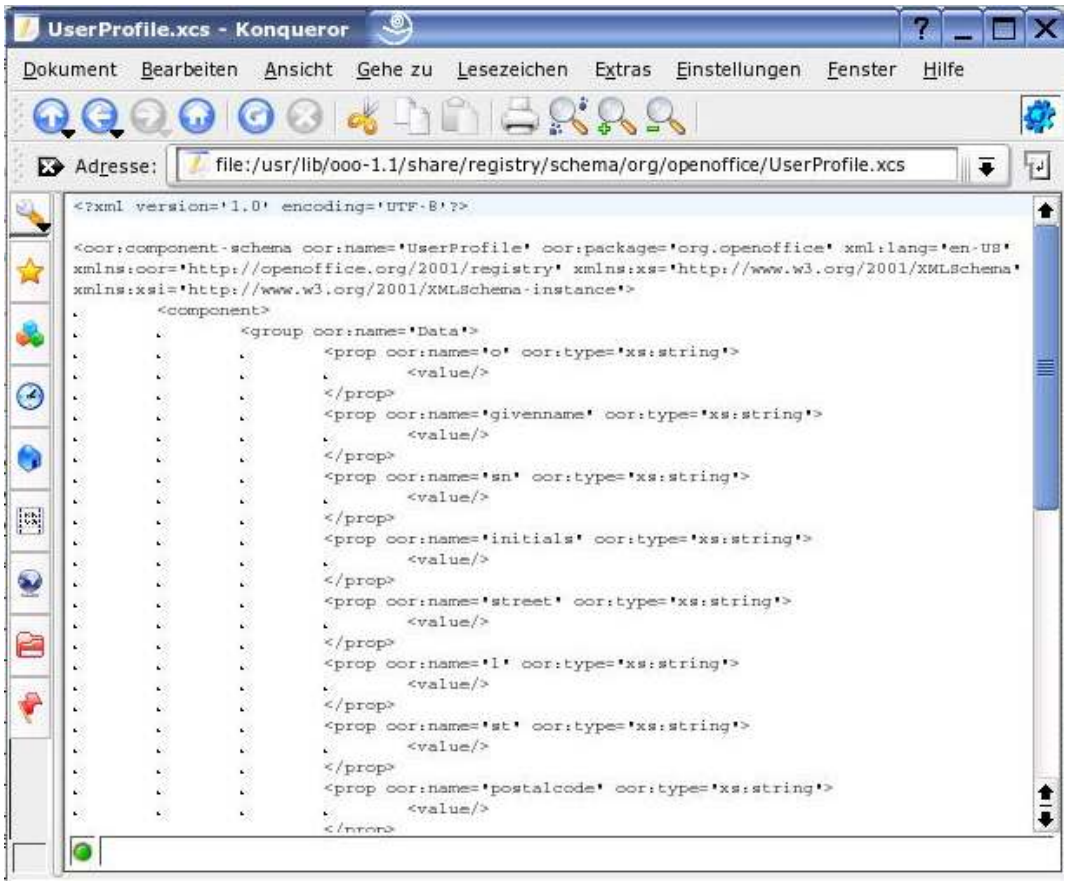


Abbildung 17: Eine XML-Konfigurationsdatei für Openoffice.org

3.3.2.2 Bewertung der Konzepte

Die Bewertung der Konfigurationskonzepte, die im Bereich Linux angewendet werden, gestaltet sich auf dem ersten Blick schwierig, da die Frage auch die grundlegende Diskussion um Windows oder Linux streift. Linux ist ein Paradebeispiel für die Anwendung von einfachen Textdateien für das Hinterlegen von Einstellungen. Diese Grundeinstellung, „Everything is a file“ wird seit seiner Entstehungsgeschichte von Entwicklergruppen mit unterschiedlicher Zielstellung verfolgt. Da diese Unterschiede auch nicht in Zukunft weniger werden, ist es nicht abzusehen, dass es bei Linux ein einheitliches System geben wird. Einerseits forciert diese Einstellung ein größeres Forschungsfeld, anderseits gibt diese Entwicklung weniger Raum für ein allgemeingültiges Gesamtkonzept.

Diese Grundeinstellungen haben aber einen entscheidenden Vorteil gegenüber Windows, der durch die beiden Beispielabbildungen des KDE beispielhaft dargestellt wird: Jede Änderung des Systems, egal ob per Konsole, grafische Oberfläche oder direktem Eingriff in die Konfigurationsdatei ist logisch und schnell nachvollziehbar. Das ist zum einem sehr ansprechend, da alles schnell per Hand geändert werden kann, zudem ist es, wenn es sich nicht um Systemdateien handelt, nicht so riskant wie ein Eingriff in die Windows Registry. Zum anderen muss

man diese Dateien erst einmal im Dateisystem auffinden und erkennen, in welchem Format, also XML, Registry- oder C-ähnlich, sie geschrieben sind. Die Entwickler von Anwendungssoftware haben es unter Linux vergleichsweise einfach. Schnittstellen zu anderen Programmen oder Bibliotheken sind, wie jeder Quellcode auch, bekannt und können verwendet werden, um systemweite Daten zu nutzen. Eine Zusammenfassung der Vor- und Nachteile linuxbasierter Konfigurationskonzepte:

Bewertung linuxbasierter Konfigurationskonzepte[158]	
Vorteile	**Nachteile**
• Einfache Konfigurierbarkeit für Nutzer und Entwickler durch bekannte Strukturen und damit eine höhere Stabilität als Windows • Leichte Nachvollziehbarkeit • Mehrbenutzerbetrieb und Netzwerkfähigkeit • Hohe Unterstützung durch Werkzeuge • Eröffnet den Weg zu innovativen Lösungen • Schnelle Entwicklungszyklen durch eine große Entwicklergemeinde • Reduzierte Fehleranfälligkeit, da Daten relativ unabhängig sind • Einheitliche Speicherung der Systemdaten • Alle Konfigurationsdateien sind in den Manuals und How-To Anleitungen ausführlich beschrieben • Das Rechtesystem von Linux führt zu einer hohen Sicherheit auch bei den Konfigurationsdateien	• Auffindbarkeit und Identifizierung der Daten • Redundanz • Aktualisierung aller Daten begrenzt möglich • Generell gilt: es gibt wenig (Business)-Software und wenig Treiber für Linux • Installationen und Einrichtung von System ist in der Regel komplexer als bei Windows

Tabelle 26: Bewertung linuxbasierter Konfigurationskonzepte

3.4 Ergebnisse der Analyse von Konfigurationskonzepten

3.4.1 Zusammenfassung der Ergebnisse

Vorgestellt wurden Konzepte zur Entwicklung von Systemfamilien, die für den flexiblen Einsatz von wiederverwendbaren Komponenten für die Ableitung von Systemfamilien verwendet werden können. Dazu gehörten Paradigmen, Konzepte, Tools und Architekturen, deren Vorgehensweise und Anwendbarkeit im Laufe des Softwareentwicklungsprozesses erläutert wur-

[158] Vgl. [Kofl 2003] S. 456 ff und [PCFo 2005] 1642 und [LiFo 2005] 117571

den. Hier wurden Vor- und Nachteile aus der Literatur zusammengefasst. Dazu kamen standardmäßig verwendete Konzepte des Konfigurationsmanagements von Windows und Linux.

Für das FORE-Modell kann aus den Analyseergebnissen abgeleitet werden, dass von Anfang an eine konsistente Modellierung der Anforderungen für grundlegende Architekturentscheidungen in den ersten Phasen des Entwicklungsprozesses wichtig ist, da komponentenbasierte Systemfamilien nicht aus traditionellen Softwareprojekten abgeleitet werden können. Die Verwendung von Komponenten spart dabei Zeit und Geld und kanalisiert Wissen vorhergehender Programmentwicklungen. Sauber gestaltete Systeme lassen sich auch später besser warten, erweitern und modifizieren. Generatoren, die z.B. auf GenVoca basieren, fokussieren dabei die vollständige Ableitung von Familienmitgliedern und benötigen dafür ein konsistente regelbasierte Grammatik. Deswegen sollten von Anfang an bei Abschluss der Anforderungserhebung verschiedene Sichtweisen und Zusammenfassungen nicht nur aus herkömmlicher funktionaler Sicht erfolgen, sondern auch andere Gesichtspunkte wie Aspekte, Funktionsgruppen, Schichten oder Architekturgrundgerüste modelliert werden, um so die sich anschließende Entscheidung für die Grobarchitektur und die zu verwendenden Technologien zu erleichtern und für die Implementierung bereits einen Grundstein für die Vorgehensweise des Programmentwurfs zu schaffen. Die Entscheidung für die Entwicklung von Systemfamilien sollte bereits in der Planungsphase feststehen. Des Weiteren hat dieses Kapitel aufgezeigt, welche Konzepte für die Hinterlegung von Software in den beiden meist verbreiteten Betriebssystemen zur Laufzeit verwendet werden. Generell kann man davon ausgehen, dass jedes Problem hier seiner spezifischen Lösung bedarf. Ein einfaches Anwendungsprogramm, welches keinerlei Kommunikation mit anderen Programmen benötigt, ist mit der einfachen und übersichtlichen Struktur einer Textdatei gut beraten. Dagegen bedarf die Mächtigkeit eines Betriebssystems eines leistungsstarken, wie auch komplexen Aufbaus einer Datenbank. Auch vom Sicherheitsaspekt her ist es sinnvoller, systemweite Einstellungen versteckt und verschlüsselt zu hinterlegen. Die Verschlüsselung kann bei linuxbasierenden Softwareprojekten jedoch nicht angewendet werden, da diverse Lizenzen auch bei weiterentwickelter Software dies verbieten.

Sinnvoll im Zusammenhang mit der Erstellung von Systemfamilien wäre die Entwicklung eines Kataloges ähnlich der Bestrebungen der FH Mannheim, in dem über bestimmte Kriterien in einem Modell die richtigen Hilfsmittel ausgewählt werden können. Hierzu müsste das entsprechende Wissen gesammelt und aufbereitet werden. Dies könnte z.B. mit Hilfe von erweiterten und bewerteten Merkmalsmodellen für die Katalogisierung der Softwareanforderungen und Systemeigenschaften geschehen.[159]

[159] Vgl. [Arno 2005] Folie 56 ff

Die hier vorgestellten Konzepte seien im Folgenden zusammengefasst:

Gesamtbewertung									
Zuordnung	**Methode/ Technik**	**Phasen-zuordnung**	**Bindungs-zeitraum**	**Funktionalität**	**Zuverlässigkeit**	**Benutzbarkeit**	**Effizienz**	**Änderbarkeit**	**Übertragbarkeit**
Paradigmen	GP	Entwurfsphase	CZ, LZ	+	o	o	o	–	+
	AOP			o	o	o	+	o	o
Schichten-architekturen	GenVoca		CZ	o	+	o	o	o	+
	Frameworks		CZ	+	+	o	o	o	o
	Plugins		LZ	+	+	o	+	o	+
Programmier-konzepte	Entwurfsmuster	Entwurf und Implementierung	CZ, LZ	o	+	o	o	o	o
	Polymorphismus			o	+	+	o	o	o
	Ifdef-Anweisungen			o	o	+	o	–	–
Komponenten-architekturen	COM / ActiveX	Implementierungs-phase	LZ, PS	+	–	+	o	o	–
	JavaBeans			+	+	+	+	o	+
	CORBA			+	o	+	o	o	+
Konfigurati-ons-DB	Win Registry	Abnahme- und Einführungsphase	LZ, In, PS	+	o	–	o	o	o
	Linuxbasierende DB			o	o	+	o	+	o
Konfigurati-onsdateien	Win INI-Dateien			o	–	o	+	+	o
	Linux conf-Dateien			o	–	o	+	+	o

CZ – Compilerzeit
LZ – Laufzeit
In – Installation
PS – Programmstart

\+ wird stark unterstützt
o wird unterstützt
– wird weniger unterstützt
(im Vergleich)

Tabelle 27: Gesamtbewertung

Dabei wurden die Kriterien in Anlehnung an die DIN ISO 9126 verwendet. Dies umfasst[160]:

- **Funktionalität** als Sammelbegriff für den Umfang der angebotenen Funktionen, der Richtigkeit der Ergebnisse und die Verträglichkeit mit anderen Systemen oder Komponenten und der Eignung für die Praxis.
- **Zuverlässigkeit** als Merkmal der korrekten Arbeitsweise bei normalen und erschwerten Bedingungen, dem Aufwand bei Wiederaufnahme der Lauffähigkeit und Reaktion bei Verfälschung oder Datenverlust.
- **Benutzbarkeit** als Kriterium für die Verständlichkeit, Handhabbarkeit und Erlernbarkeit des Konzeptes.
- **Effizienz** als Zeitdauer und Ressourcenbedarf wie Speicherplatz und Prozessorleistung bis zum Ergebnis.
- **Änderbarkeit** als Maß für die Wartbarkeit, Durchführbarkeit, Verständnisdauer des Programms, Stabilität und Überprüfbarkeit des Konzeptes.
- **Übertragbarkeit** als Kennzeichen für den Aufwand der Implementierung, für die Anpassbarkeit an andere Umgebung und für die Schnittstellenverträglichkeit.

Aus den Ergebnissen dieser Tabelle geht hervor, dass alle Konzepte und Methoden mehr oder weniger zu der Softwarequalität beitragen. Oft werden aufgrund der Komplexität des Konzeptes Abstriche gemacht. Trotzdem kann nicht allgemeingültig gesagt werden, dass sich gewisse Konzepte schlechter eignen, sondern einfach für andere Projekte geeignet sind. So reduziert die Mächtigkeit und Fehlerintoleranz einer Windows Registry die Benutzbarkeit, erhöht aber die Funktionalität umso mehr. Die Generative Programmierung dagegen bietet unter Einbußen der Benutzbarkeit und Änderbarkeit einen hohen Funktionsumfang und lässt sich einfach übertragen. Komponenten- und Pluginarchitekturen unterstützen überdurchschnittlich viele Kriterien, und werden auch immer häufiger aufgrund ihrer flexiblen Struktur eingesetzt.

[160] Vgl. [Balz 2001] S. 1113 und [Phil 2005] S. 46 ff

3.4.2 Ergebnisse für die Konfiguration der VDR-Software

Für die Konfiguration der Live-CD wird anhand der Analyseergebnisse und den bereits vorhandenen Plugins die Weiterführung des Pluginmodells verfolgt. Die Konfigurationsdaten selbst werden dabei in einzelnen unverschlüsselten Textdateien vorgenommen, die mit dem XML-Format ausgedrückt werden, da das Projekt einen relativ geringen Umfang hat. Bei größeren Projekten würden die positiven Aspekte einer datenbankorientierten Konfiguration überwiegen. Das XML-Format wird auch deshalb gewählt, da die Anforderungen und Eigenschaften des DVP-Projekts bereits in XML vorliegen. Zudem bietet XML eine Unterstützung dynamischer Inhalte, bessere Lesbarkeit und große Unterstützung auf internationaler Ebene, da XML das Datenaustauschformat der Zukunft ist.[161]

[161] Vgl. [Micr 2005-2]

4 Beispiel im Rahmen des DVP-Projektes

Für die praktische Demonstration der in Kapitel 3 vorgestellten Konfigurationsmöglichkeiten soll eine bestehende Live-CD untersucht werden, die eine kundenspezifische Anordnung der Plugins der VDR-Software enthält.

Dafür wird die von Melanie Schöppe und Martin Rulsch entwickelte Demo-CD im Rahmen ihrer gemeinsamen Studienjahresarbeit „Digitales Video Projekt auf einer Boot-CD“[162] verwendet und analysiert. Die prototypische Live-CD wurde ausschließlich für die Testumgebung im Labor der TU Ilmenau erstellt. Dabei soll die Softwarequalität mittels geeigneter vorgestellter Methoden und Techniken im Sinne eines Software Reengineerings und Refactorings erhöht werden, um zukünftige Änderungen leichter realisieren zu können.

Refactoring als ein Teilbegriff des Reengineerings hat das Ziel, die Softwarequalität ohne Änderung der eigentlichen Funktionen hinsichtlich der Wiederverwendung, Nachvollziehbarkeit, Wartbarkeit, Anpassbarkeit, Lesbarkeit und Optimierung der Ressourcennutzung zu verbessern und für zukünftige Systeme Schwachstellen zu finden und zu eliminieren. Im Zuge des Reengineerings können dann neue Anforderungen schneller eingepflegt werden. Beide Begriffe werden jedoch oft gleichbedeutend verwendet. Historisch gesehen gründet sich die Softwaresanierung mit Funktionsäquivalenz in den 1980er Jahren, als man damit begann, Software zu strukturieren und zu modulieren. Das sollte die Lesbarkeit erhöhen und Operationen auslagern. Heute liegt der Fokus auf der Wiederverwendbarkeit von Softwareteilen. Die Objektorientierung bietet sich dabei an, Klassenstrukturen sind jedoch zu feingranular, so dass Wiederverwendung in gröberen Strukturen wie Komponenten stattfinden muss.[163]

Das Reengineering von Software als Überbegriff umfasst geeignete Methoden und Techniken, sowie alle Schritte, die die qualitative Verbesserung, Aufbereitung und Evolution von Programmen unterstützen. Möglich ist dies in folgenden Bereichen:[164]

- **Emergency Repairs**: unverzügliche Behebung von Fehlern
- **Korrektive Systemerhaltung**: Behebung von Fehlern im nächsten Release
- **Erweiterung / Adaption**: Anpassung an geänderte oder neue Anforderungen
- **Migration**: Überführung in eine neue Umgebung

[162] Vgl. [Schö 2005]
[163] Vgl. [Snee 2001] S. 5
[164] Vgl. [Kuhl 2004] Home

- **Integration**: Einbindung in umfassendere Systeme
- **Sanierung**: Verbesserung der Software-Qualität ohne Änderung der Funktionalität
- **Re-Dokumentation**: nachträgliche Erstellung oder Verbesserung der Dokumentation

So können z.B. Maßnahmen im objektorientierten Code vorgenommen werden:

- Umbenennen von Methoden
- Umbenennen von Klassen
- Verschieben von Methoden
- Aufteilen von Klassen
- Zusammenfassen von Klassen

Die genannten Ziele und Maßnahmen der Zielerreichung sollen auf die bestehende Live-CD angewendet werden. Dazu wird im Folgenden Unterkapitel die CD im Rahmen einer Ist-Analyse vorgestellt. Des Weiteren werden anhand der Analyseergebnisse des 3. Kapitels theoretische Vorschläge im Sinne des Refactorings und Reengineerings gemacht, um das Konfigurationsmanagement zu verbessern.

Der Aufbau der Pluginarchitektur kann folgender Abbildung entnommen werden:

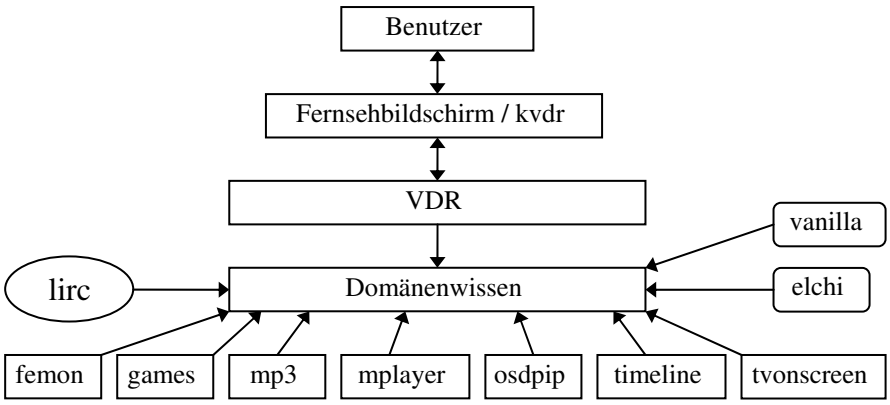


Abbildung 18: Aufbau der Pluginarchitektur im Beispiel

4.1 Ist-Analyse und Domänenwissen

Im Folgenden wird die bestehende CD beschrieben, um mögliche Ansatzpunkte für ein Refactoring zu erfassen.

4.1.1 Beschreibung der CD

Im Labor wird nachstehende relevante Hardware verwendet:[165]

CPU: Intel Pentium 4 1500 MHz
Festplatte: 40 GB von Maxtor
Grafikkarte: ATI Technologie Inc. Radeon RV 100
DVB-Karten: Technotrend DVB Card „Budget S" und DVB Card „Premium"

Der vollständige grafische Aufbau kann dem Anhang A entnommen werden.

Die vorliegende CD besteht aus einer remasterten Knoppix 3.7 Version mit einem 2.6 Kernel, in der diverse für die Demo-CD nicht notwendige Programme deinstalliert wurden. Dazu kamen die aktuellen DVB-Treiber und der Xfce 4 Desktop[166], der wesentlich kleiner und schneller ladbar ist als das von Knoppix standardmäßig zugehörige KDE. Zusätzlich wurde der Apache-Server 2.0 mit dem Python Modul 3.1 sowie dem Python-Programm 2.3 eingerichtet. Dazu wurde VDR inklusive dem Patch[167] Elchi installiert. Der zusätzliche Patch Vanilla kann optional gewählt werden.

[165] Vgl. [Schö 2005] S. 6

[166] Mehr dazu unter [Xfce 2005]

[167] Ein Patch greift direkt in eine Programmlogik ein, da es direkt den Quelltext beeinflusst und die gesamte Nutzerschnittstelle der Software ändert.

Zu Demozwecken wurden folgende Plugins[168] installiert:

Bereits integrierte Plugins der Demo-CD[169]	
Name	**Kurzbeschreibung**
Plugin femon 0.07 11.09.2004	Darstellung der DVB Signal Informationen des aktuellen Senders auf den OSD.
Plugin games 0.6.1 23.05.2004	Sammlung von Spielen: Snake, Tetris, Tron, TicTacToe, Minesweeper.
Plugin mp3 0.9.9 16.01.2004	Wiedergabe von mp3s, WAVs etc. Das Bedienkonzept ist Playlisten-zentriert.
Plugin mplayer 0.9.9 16.01.2004	Spielt verschiedene Videoformate wie z.B. (S)VCDs und DVDs ab.
Plugin osdpip (picture in picture) 0.07.1 26.10.2004	Zeigt gleichzeitig 2 Kanäle auf dem OSD, ein Kanal aus Vollbild, der andere als verkleinertes Bild in einer Ecke. (gleicher Transponder, sonst sind 2 DVB-Karten nötig)
Plugin timeline 0.9.0 02.01.2005	Grafische Darstellung vorprogrammierter Aufnahmen und deren Konflikte im Tagesverlauf.
Plugin tvonscreen 0.7.0 02.01.2005	Anzeige von EPG-Daten von 3 Fernsehsendern in einer TV-Zeitschriften-ähnlichen Ansicht.

Tabelle 28: Plugins der Demo-CD

Die beispielhaften Plugins wurden aus visuellen Gründen für die Demo-CD gewählt. Die jeweilige Bedienanleitung der Plugins ist der Studienjahresarbeit [Schö 2005] S. 8 ff zu entnehmen. Das fertige System lässt sich in der Laborumgebung ausführen. Die bootbare CD startet automatisch eine Weboberfläche im Warenkorbdesign, in dem Plugins, Patchs und die Bedienung ausgewählt werden können und sich dann die konfigurierte VDR-Software starten lässt. Die Bildausgabe kann über den angeschlossenen Fernseher oder über den installierten Player kvdr gesteuert werden. Die Bedienung erfolgt wahlweise per Tastatur oder Fernbedienung. Dazu wurde das Modul lirc[170] eingerichtet.

[168] Ein Plugin erweitert ein Programm lediglich um zusätzliche Funktionen.

[169] Vgl. [Schö 2005] S. 8 ff und [VDR-W 2005] Plugins

[170] Lirc ist ein Projekt, welches das Programm über eine Infrarotschnittstelle steuern kann. Lirc enthält Gerätetreiber für den Infrarotempfänger und stellt die Signale der Anwendung zur Verfügung.

Die Ansicht der Warenkorboberfläche:

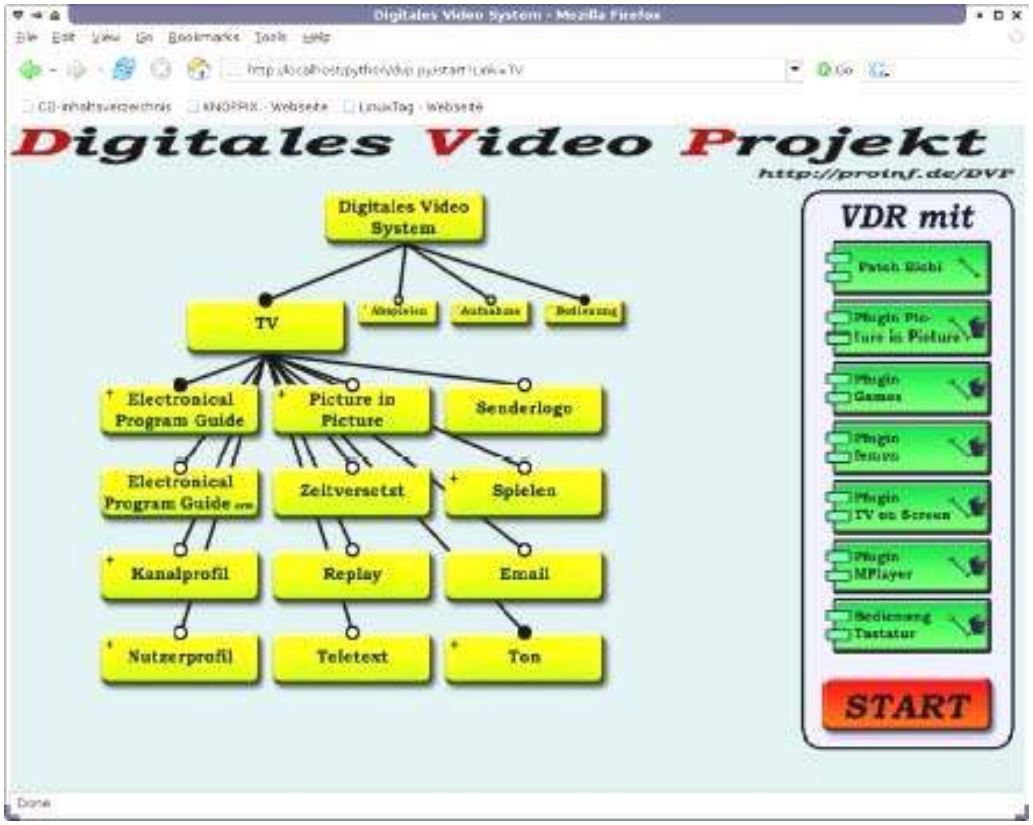


Abbildung 19: Ansicht des Menüs der bestehenden CD

Über Konfigurationsdateien mittels XML-Fragmenten wird die Weboberfläche dynamisch erstellt. Beim Drücken des Startbuttons wird dann die zusammengesetzte Anweisung an ein Terminal geleitet, das den VDR mit den gewünschten Plugins startet.

Die Vorgehensweise sei hier grafisch dargestellt:

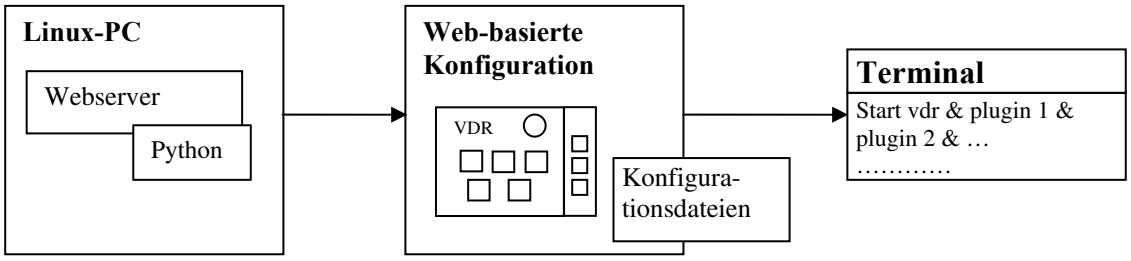


Abbildung 20: Vorgehensweise der Demo-CD

Die komplette Notation der FORE-Anforderungen und –Merkmale kann dem Anhang B entnommen werden. Die Notation wird in dem Menü widergespiegelt; realisierte Anforderungen der CD wurden im Programm mit einem „+“ markiert und enthalten jeweils einen weiteren Knoten. Mögliche Auswahl in der ersten Ebene:

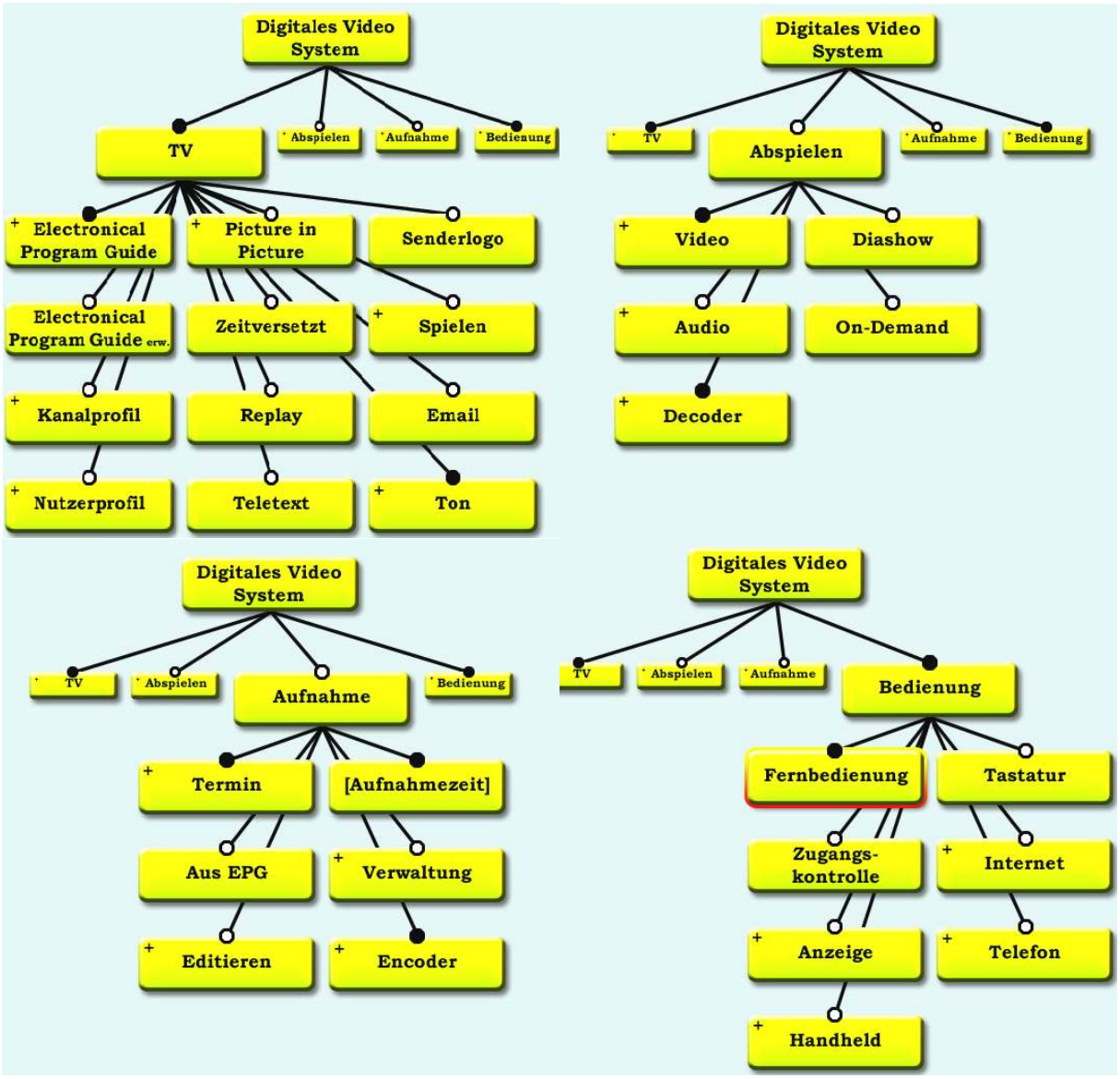


Abbildung 21: Umsetzung der Anforderungen als FORE-Notation

Um die Verzeichnisstruktur im Auge zu behalten, seien die für VDR relevanten Pfade hier zusammengefasst:

Relevante Pfade der Demo-CD	
Pfad	**Erläuterung**
var/www/python var/www/Plugins var/www/bilder.org /etc/vdr/Plugins /usr/lib/vdr/plugins/elchi oder vanilla	Python Programme XML Fragmente Grafiken Konfigurationsdateien der Plugins Bibliotheken

Tabelle 29: Pfade der Live-CD

4.1.2 Eingriffsbereiche

Bereiche, in denen Änderungen vorgenommen werden können, werden im Folgenden beschrieben.

4.1.2.1 Bootvorgang

Da der Bootvorgang und die Hardwareerkennung Knoppix-eigen sind, sind Änderungen hier nicht für das VDR-Programm nötig bzw. nicht immer möglich. Jedoch muss hier erwähnt werden, dass aufgrund von Kapazitätsbeschränkungen eine große Zahl an Software und Treiber aus der CD und dem Kernel entfernt wurden. So reduziert sich der Umfang der Demo-CD weiterhin auf die Größe einer CD, lässt sich jedoch ausschließlich für die Laborumgebung nutzen. Deswegen wäre es angebracht, bei weiteren Versionen den vollen Kernelumfang der Knoppix-Version bestehen zu lassen und anschließend eine DVD zu brennen, um so die volle Hardwareerkennung und die Treiberaktivierung zu nutzen.

4.1.2.2 Die Weboberfläche

Die Weboberfläche wird durch den laufenden Apacheserver gesteuert. Der Nutzer selbst sieht lediglich die lokale Webseite, die von dem Server übergeben wird, nachdem das Programm dvp.py durch die Nutzereingaben die Seiten aus den HTML- und XML-Fragmenten und den Bildern zusammengesetzt hat. Dazu gehört die entsprechende CSS-Datei, die das Aussehen der Webseite steuert. Die HTML-Fragmente enthalten jeweils Teile zur Überschrift, zur Navigation und zum Warenkorb mit Platzhaltern für die Grafiken und Links, die dann jeweils mit den XML-Fragmenten gefüllt und je nach Pluginauswahl zusammengesetzt werden. Für die XML-Dateien gibt es 3 verschiedene DTDs: für die Plugins selbst, für das Bedienungsmodul und für die Steuerung des Ganzen.[171]

[171] Vgl. [Schö 2005] S. 42 ff

Der genaue Ablaufplan ist der folgenden Abbildung zu entnehmen:

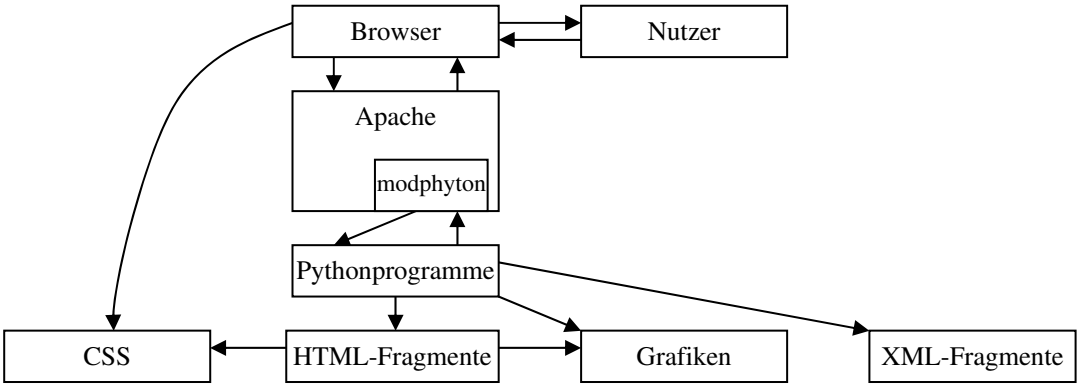


Abbildung 22: Zusammenspiel von Apache, Python und dem Menü[172]

4.1.2.3 Der Programmcode

Das Skript dvp.py setzt aus den einzelnen Teilen die HTML-Seiten für die Weboberfläche zusammen, gibt sie zurück an den Webserver und startet und stoppt den VDR. Im Quelltext selbst werden die entsprechenden Pfade am Anfang deklariert. Eine Veränderung ist hier leicht und verständlich möglich. Python lässt sich sowohl prozedural als auch objektorientiert verwenden. Das hier verwendete Programm ist jedoch nicht objektorientiert geschrieben.

Deswegen besteht der Quellcode aus diversen Funktionen, die folgendermaßen miteinander agieren:

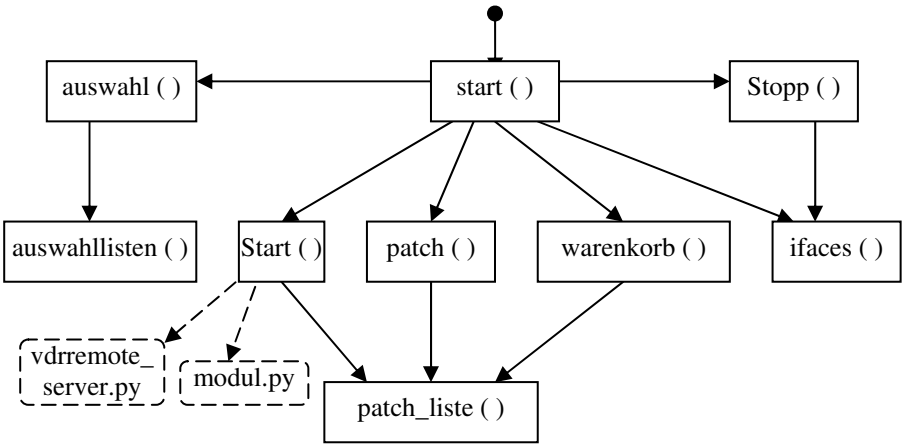


Abbildung 23: Prozedur- und Programmaufrufe im Hauptprogramm[173]

[172] Vgl. [Schö 2005] S. 44
[173] Vgl. [Schö 2005] S. 51

Die einzelnen Funktionen haben kurz zusammengefasst folgende Aufgaben:

- **start ( ):** initialisiert das Menü, verteilt die HTML-Anfragen auf die anderen Funktionen, Sessionmanagement
- **ifaces ( ):** ermittelt IO-Ports und IRQs für die serielle Schnittstelle (Fernbedienung)
- **auswahl ( ):** regelt die Liste ausgewählter Plugins, Steuerung und Kategorieansichten
- **auswahlisten ( ):** Regelt die Scrolldown-Auswahlmenüs der Fernbedienung
- **patch ( ):** Regelt das Scrolldown-Auswahlmenü für die Patchs
- **warenkorb ( ):** Erstellt den HTML-Code für den Warenkorb aus 3 Fragmenten
- **Start ( ):** setzt die Start-Kommandozeile für den VDR unter Verwendung der Hilfsprogramme vdrremote_server.py und modul.py zusammen
- **Stop ( ):** beendet den VDR, gibt die Schnittstelle von lirc wieder frei

4.1.2.4 Integration neuer Plugins

Trotz einer detaillierten Beschreibung in der Studienjahresarbeit[174] ergeben sich bei der Installation eines Plugins 5 praktische Probleme:

1. Da die Ordnerstruktur nicht den Standardvorgaben entspricht, müssen die Makefiles und die Includedateien umgeschrieben und ein neues Debian-Paket für das Plugin erstellt und kompiliert werden.
2. Da die beiden Patchs je einer Installation bedürfen, sind dabei für jedes Plugin 2 Debian-Pakete für den gepatchten VDR nötig.
3. Für das Einbinden des Plugins in das Programm müssen neue Grafiken für die Navigation, den Warenkorb und die Buttons für das gesamte Menü erstellt werden, was trotz Templates mit höherem Aufwand verbunden ist, da die Bilder nicht dynamisch erstellt, sondern statisch geladen werden.
4. Dazu muss zu jedem Plugin mindestes ein HTML-Fragment geändert oder erstellt, und mindestens ein XML-Fragment erstellt werden.
5. Außerdem kann ein Plugin auch Zusatzkonfigurationen, und damit zusätzliche HTML-Elemente wie z.B. eine Scrollbox, oder andere Plugins erfordern.

Die Probleme bei der Integration können durch nachfolgende theoretische Änderungen im Programm selbst und der Konfiguration der Weboberfläche verbessert werden.

[174] Vgl. [Schö 2005] S. 55 ff

4.2 Soll-Konzept: Mögliche Änderungen

Mögliche Änderungen sind auf verschiedenen Ebenen möglich. Zum einen können Änderungen im Code bzw. im Aufbau des Gesamtprogramms die Softwarequalitätskriterien erhöhen, zum anderen lässt sich auf Ebene der Weboberfläche das Kriterium der Benutzbarkeit anhand der DIN ISO 9241-10 weiter spezifizieren. Integrativ lässt sich damit das Verfahren zum Einfügen neuer Plugins erleichtern.

4.2.1 Änderungen auf Code-Ebene

Das prototypische Programm für die hier verwendete und pluginbasiert konfigurierte VDR-Software ist nicht objektorientiert geschrieben. Die prozedurale Aufteilung des Programmcodes und die Verwendung von Templates sind aber bereits grundlegende Schritte zur Wiederverwendung von Teilen, nutzt aber noch nicht den vollen Umfang, den die Objektorientierung bietet. Die Hauptziele sind hier die Erhöhung der Softwarequalität, Wiederverwendbarkeit und die Reduzierung des Aufwands, der nötig ist um ein weiteres Plugin zu integrieren, z.B. durch bestimmte Automatisierungen. So ergeben sich in erster Linie folgende Überlegungen:

Allgemeine Änderungen auf Code-Ebene
• Präzisere und aussagekräftigere Bezeichnung der Prozeduren • Zusammenfassung der XML-Fragmente in ein einheitliches Dokument • Entwicklung eines allgemeingültigen Schemas für XML-Dateien • Einheitliche automatische Generierung eines dynamischen HTML-Gerüsts • Dynamisches Erstellen der Grafiken • Flexibleres Gestalten der CD um das Programm für eine Migration bzw. Integration in andere Umgebungen (vgl. Kapitel 4.1.2.1) • Änderungen der Verzeichnisstruktur, um eine Doppelinstallation der Plugins zu vermeiden (vgl. hierzu Tabelle 29) • Erweiterung um weitere Plugins, um z.B. Aufnahmen tätigen zu können, mögliche Plugins sind Anhang C zu entnehmen • Integration der externen Programmteile in das Hauptprogramm dvp.py • Reduzierung des Aufwands einer Pluginintegration • Re-Dokumentation

Tabelle 30: Änderungen des Programm-Codes

Die in Kapitel 3 vorgestellten objektorientierte Programmierkonzepte helfen dabei, das Programm umzuwandeln und so die Wiederverwendung und Wartung zu verbessern, und in einer späteren Version im Sinne des Generativen Programmierens mit Hilfe eines Generators auto-

matisch Varianten ableiten zu können. Dazu muss das bestehende Merkmalmodell in eine Form überführt werden, die ein automatisches Ableiten von Familienmitgliedern ermöglicht.

Zur besseren Strukturierung des objektorientierten Programms lassen sich dann die Vorteile der Objektorientierung wie z.B. Vererbung nutzen, um die Ziele der Systemfamilienentwicklung zu realisieren. Abzuwägen bleibt allerdings der Aufwand und der Nutzen der Umstrukturierung bei kleinen Programmen. Anwenden lassen sich so folgende in Kapitel 3.2 vorgestellte Konzepte:

Polymorphismus: Polymorphismus ist wie in jeder objektorientierten Sprache eine Standardaufgabe. Python stellt dazu eine Vielzahl an polymorphen Mitteilungen zu Verfügung und gibt die Möglichkeit eigene Mitteilungen zu erstellen. Denkbar wäre in der praktischen Anwendung hier z.B. die Zuordnung einer Methodenanwendung für Start und Stopp auf ein Plugin zur Laufzeit.[175]

Entwurfsmuster: Muster lassen sich auch in Python verwenden oder können hier neu entwickelt werden. Denkbar wäre es z.B. im neu strukturierten Programm ein Klassenkonstrukt nach dem Muster des „Prototypen“ zu schaffen, welches die Grafikdarstellung, den HTML-Aufbau oder die XML-Generierung neuer Plugins regelt, denn das Muster „Prototyp“ hat die Aufgabe durch die Verwendung eines prototypischen Exemplars neue Objekte durch Kopieren zu erzeugen.[176]

Aspektorientierte Programmierung: Als orthogonale Zusammenfassung von technischen Details neben der Semantik regelt die Aspektorientierte Programmierung Rechteüberprüfungen, Persistenz, Synchronisation von Daten, Objektinteraktionen, Parameterübergaben etc. und erhöht damit die Wiederverwendbarkeit der „gefilterten“ Codeteile enorm. Denkbar wäre hier im praktischen Beispiel z.B. eine Routine, die vor Programmstart prüft, ob für die Firmware oder für die VDR-Software und ihre Plugins Updates verfügbar sind.[177]

[175] Vgl. [Lani 2000] S. 231 ff

[176] Vgl. [Gamm 1996] S. 144 ff

[177] Für das modpython des Apacheservers gibt es inzwischen ein aspektorientiertes Framework namens Pylets. Mehr dazu unter [OSTG 2004]

4.2.2 Änderungen auf Seiten des Webmenüs

Um eine Benutzeroberfläche von Anwendungsprogrammen aufgrund ihrer Ergonomie zu bewerten, wurde die ISO 9241-10 (1996) „Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 10: Grundsätze der Dialoggestaltung. Brüssel: CEN“ ins Leben gerufen. Sie beurteilt die Software-Ergonomie für benutzerfreundliche Dialoggestaltung nach folgenden Gesichtspunkten unabhängig von der Dialogtechnik:[178]

- **Aufgabenangemessenheit:** Komplexität der Bedienung, Funktionsumfang zur Bewältigung der Anforderungen
- **Selbstbeschreibungsfähigkeit:** Bietet Überblick über das Funktionsangebot
- **Steuerbarkeit:** Welche Funktionen sind in welcher Form enthalten
- **Erwartungskonformität:** entspricht das Programm kognitiven Erwartungen
- **Fehlerrobustheit:** wie tolerant ist das System gegenüber Bedienfehlern
- **Individualisierbarkeit:** wie lässt sich das System an eigene Wünsche anpassen
- **Lernförderlichkeit:** inwiefern hilft die Oberfläche dem kognitiven Verständnis des Nutzer

Nach diesen Gesichtspunkten ergeben sich für die Weboberfläche des Programms folgende Vorschläge, die einerseits die Ergonomie erhöhen und gleichzeitig auch hier den Integrationsaufwand eines neuen Plugins reduzieren:

[178] Zitiert nach [Rich 1999] S. 1 ff

Vorschläge für das Menü zur Erhöhung der Ergonomie[179]
• Hinterlegung des Maulschlüssels mit Konfigurationsmöglichkeiten von Optionen und Parametern per Auswahllisten etc. (siehe Modul Fernbedienung) • Vorgabe von Tastenkombinationen für Fernbedienung und Tastatur • Vorgabe von definierten Channel.confs für verschiedene Gegenden • Lernumgebung für Tastenkombinationen vor Start des Programms per grafischer Oberfläche und / oder direktem Tastendruck • Button für die Erstellung einer Kunden-CD, die den bereits fertig konfigurierten VDR automatisch startet, dies kann auch nach verschiedenen Gesichtspunkten generiert werden, z.B. ein VDR spezialisiert auf Wiedergabe von Aufnahmen, zum optimalen Aufnehmen mit Timeshift, Timerfunktionen etc., oder einem VDR-Multimediacenter, welches sämtliche Formate abspielen kann • Einrichtung der Nutzung der CD unter Analogkarten mittels Analog-Plugin • Einrichtung der Nutzung der CD unter Budgetkarten mittels Xine-Plugin[180] • Einrichtung der Nutzung der CD ohne DVB-Karte als Client eines Streaming-Servers[181] • Markierung und Nichtstarten bzw. Entfernen aus dem Warenkorb von Plugins, die von dem ausgewählten Patch nicht unterstützt werden • Anzeige möglicher Plugin-Kombinationen ähnlich einem Automobilzubehörkatalog • Hilfsfunktionen für Manuals und / oder Bedienungsanleitungen • Benutzbarkeit der Tastatur außerhalb des VDR zur Laufzeit

Tabelle 31: Änderungen der Weboberfläche zur Erhöhung der Ergonomie

[179] Vgl. [Schö 2005] S. 65 ff

[180] Das Xine-Plugin übernimmt die Decodierung des MPEG2-Stroms, die bei full-featured Karten normalerweise von der Hardware übernommen wird. Die Nutzung dieses Plugins ermöglicht die Verwendung günstiger Low-Budget Karten, die dadurch aber mehr Arbeitsspeicher benötigen.

[181] Einen Lösungsvorschlag dazu unter [Webe 2005]

4.3 Zusammenfassung der Ergebnisse

Im Laufe der Untersuchung der Demo-CD hat sich ergeben, dass diese in der Laborumgebung der TU Ilmenau lauffähig ist und sich durch den verschlankten Kernel nicht auf andere Systeme migrieren lässt. Zum anderen wäre es denkbar, statt der aufwändigen Modifizierung der CD über die Anpassung des Kernels, umfangreiche Deinstallationen und Einrichten der Xfce-Oberfläche, das Programm und die Weboberfläche auf ein bereits existierendes Knoppix-Derivat namens Xfld mit XFCE-Oberfläche[182] zu migrieren und das System auf eine DVD zu überführen, damit Platzgründe keine Rolle mehr spielen müssen. So ist das Einsatzspektrum der CD nicht zu stark reduziert. Zum anderen muss das zu Grunde liegende Programm objektorientiert angepasst werden, um die Vorteile der Objektorientierung nutzen zu können. So ist die Demo-CD ein suboptimales Demonstrationsobjekt für die Konzepte zur Generierung von Systemfamilien. In Kapitel 4.2 konnten so nur theoretische Vorschläge zur Verbesserung der Softwarequalität und zur Erhöhung der Anwendungsergonomie vorgestellt werden. Als weitere Schritte in Folgearbeiten muss die Anwendbarkeit der CD in erster Linie auf mehren Hardwareumgebungen möglich sein und das Programm sollte objektorientiert angepasst werden, um die in dieser Arbeit vorgestellten Konzepte nutzen zu können. Dann ist es möglich anhand des auf Konsistenz überprüften Merkmalmodells und eines Generators in Form eines Skriptes Varianten ableiten zu können.

[182] Dieses schlanke Knoppix gibt es unter [Xfld 2005]

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung und abschließende Bemerkungen

Das Forschungsgebiet der Softwarearchitektur gewinnt zunehmend an Bedeutung. Das systematische Strukturieren muss bereits in der Planungsphase tief verankert werden, um später Fehler zu reduzieren, die Wartung zu verbessern und Softwareteile wiederverwenden zu können, denn „Durch eine Softwarearchitektur wird nicht nur die Struktur des zu entwickelnden Softwaresystems festgelegt, auch Möglichkeiten der arbeitsteiligen Entwicklung und damit die Struktur eines Entwicklungsprojektes sind in der Architektur angelegt. Die Schnittstellen der wichtigsten Komponenten sind wesentliche Elemente der gemeinsamen Sprache der Projektmitglieder. Somit hat die Softwarearchitektur einen großen Einfluss auf das Vorgehen im Entwicklungsprojekt.“[183]

Hier haben sich komponentenbasierte Architekturen und Systemfamilien durchgesetzt, die basierend auf Vorgehensweisen reifer Technologien der Automobilbranche weg von monolithen Strukturen sich aus einem Baukastensystem flexibel und wiederverwendend bedienen. [184]

Diese Arbeit hat einen repräsentativen Querschnitt durch die Thematik der Systemfamilien gegeben. Moderne Systeme bedürfen nicht nur einer neuen grundlegenden Architektur, sondern auch einer durchgängigen methodischen Unterstützung durch angepasste Vorgehensweisen, Konzepte, Methoden und Designprinzipien, um Software erfolgreich entwickeln zu können, die Time-To-Market Zeit zu senken, Kosten zu reduzieren und die Softwarequalität für den Kunden zu erhöhen.[185]

In dieser Diplomarbeit wurde dazu in die für eine Beispielimplementierung nötige Domäne von Digitalfernsehen und entsprechender Opensource-Software, sowie Live-CDs und Systemfamilien eingeführt. Der Hauptteil der Arbeit lag in der Untersuchung der Konfigurationsmöglichkeiten entlang des Softwareentwicklungsprozesses von Systemfamilien. Dazu wurden bestimmte hier angewendete Paradigmen, Konzepte und Architekturen in den verschiedenen Phasen vorgestellt, ihre Vor- und Nachteile beschrieben und Strategien des Konfigurationsmanagements im laufenden Betriebssystem analysiert. Hierzu wurde in der Zusammenfassung eine Tabelle entwickelt, die die einzelnen Themen anhand der Softwarequalitätskriterien auf-

[183] Vgl. [Stru 2005] Prozess
[184] Vgl. [Stru 2005]
[185] Vgl. [Weis 1999] S. 19 ff

grund ihrer speziellen Ausgestaltung bewertet. Jedoch muss hier berücksichtigt werden, dass eine grundlegende Wertung nicht möglich und nicht nötig ist, da Konzepte projektspezifisch ausgewählt werden müssen und sich gegenseitig bedingen. Anhand der Analyseergebnisse wurde ein Vorgehen für das Konfigurationsmanagement der Demo-CD im anschließenden Kapitel 4 entwickelt.

In einer Zusammenführung von Digitalfernsehen, Live-CD, Opensource-Software und Systemfamilienentwicklung wurde im 4. Kapitel eine bestehende, an der TU Ilmenau entwickelte Live-CD, die eine webbasierte Kundenkonfiguration der VDR-Software für Digitalfernsehen bietet, vorgestellt, und Vorschläge anhand der Analyseergebnisse von Kapitel 3 gegeben, um im Sinne der Systemfamilienentwicklung die Softwarequalität mit Hilfe der beschriebenen Konzepte zu erhöhen. Ebenso wurde hier die Weboberfläche analysiert und Vorschläge anhand der Software-Ergonomie für ein optimiertes Nutzerinterface gegeben.

Die VDR-Software ist dabei ein innovativer Stellvertreter von Pluginarchitekturen, die sich einfach und flexibel an sich ändernde Anforderungen anpassen lassen. In der multimedialen Welt ist dies ein entscheidender Wettbewerbsfaktor, denn das Schlüsselwort der heutigen Softwareentwicklung ist dabei die Wiederverwendung. Sie entscheidet maßgeblich über den Kostenfaktor von Projekten und verbessert die Aufgabenverteilung in Softwareprojektteams.

Mit dieser Arbeit konnte gezeigt werden, auf welchem Stand sich die Entwicklung der generativen Softwareentwicklung befindet, was sie momentan durch die Methodenunterstützung leisten kann und inwiefern sie sich realisieren lässt.

5.2 Ausblick

Die Entwicklung von Komponentenarchitekturen, Systemfamilien und deren generativen Ableitung stecken noch in den Kinderschuhen, werden aber von der Industrie stark forciert. Die Befürworter sind der Meinung: „dass in näherer Zukunft die konventionelle Softwareentwicklung durch die generative ersetzt wird“186 Das Problem hierbei ist der immense Aufwand der dahinter steckt. Die Entwicklung von Software anhand ausführlicher Modellerstellung von Anforderungen und Spezifikationen auf hoher Abstraktionsebene vor der Implementierung gehört bei großen Softwareherstellern bereits zum Stand der Technik. So kann man durch die Generierung von Systemfamilien schneller Qualitätsziele erreichen.

Für kleine Projekte oder kleinere Unternehmen ist jedoch fraglich, ob sich der Aufwand auf der einen Seite lohnt, da auf der anderen Seite die Komplexität des Themas noch keine vollständige Automatisierung zulässt. So gibt es gegen die generative Softwareentwicklung noch

[186] Vgl. [Tamm 2004] S. 5

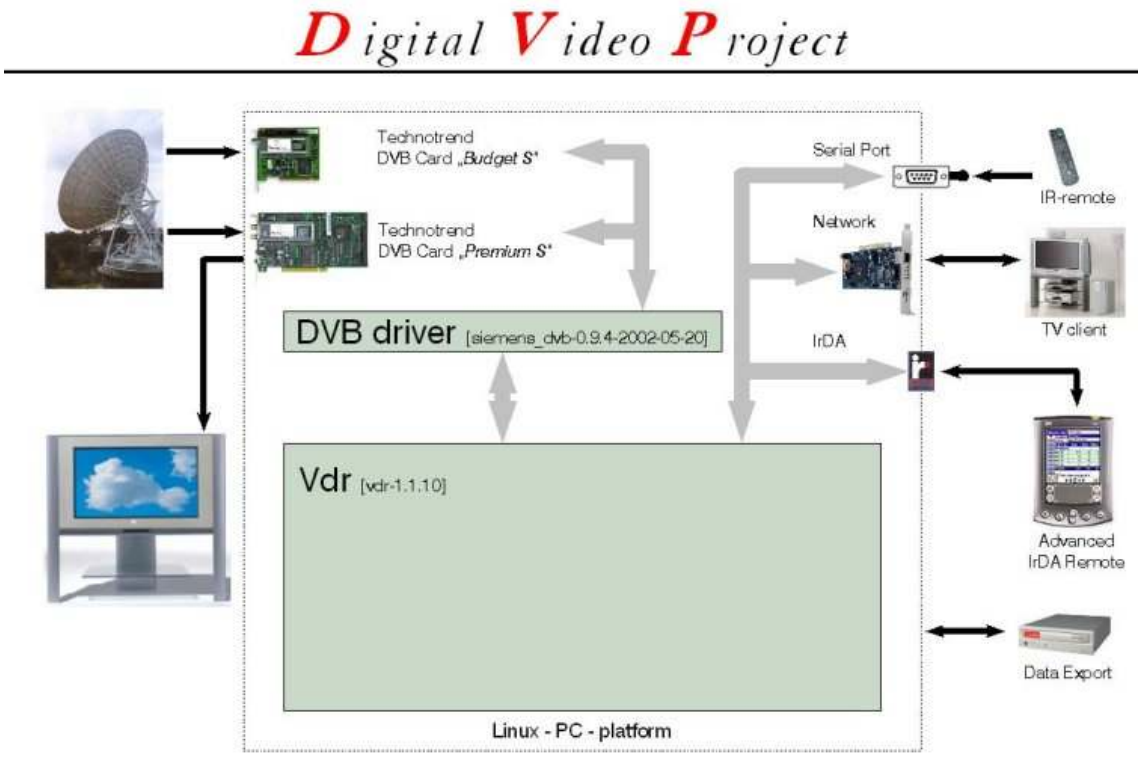
einige Hemmschwellen zu überwinden. Deswegen wurden eine Vielzahl von Förder- und Forschungsprojekten z.B. vom Fraunhofer IESE zum Thema der automatischen Generierung ins Leben gerufen[187]. Das automatische Ableiten wird, wenn auch nicht ersetzen, die konventionelle Softwareerstellung jedoch erheblich ergänzen können, da die Vorteile immens sind. Das Wissen von bereits bestehenden Architekturen wird systematisch gesammelt und weitergenutzt, Wiederverwendung wird forciert und die Wartung, Pflege und Fehlerbehebung von solchen zukünftigen Systemen wird wesentlich reduziert sein. Zudem werden Kundenanforderungen wie Kundennutzen immens steigen. Komponentenbasierte Systeme sind bereits heute sehr beliebt, der große Erfolg von solchen Plattformen wie z.B. Eclipse oder die VDR-Software zeigt dies.[188]

So soll diese Diplomarbeit mit den Worten von Krzysztof Czarnecki schließen: „The advancing technology will make it easier to package expert knowledge such that it will be possible to reuse it in a wider set of contexts, overcoming technological and domain differences. This will lead to more automation, more specialization, and will enable new kinds of applications. But there is still a long way to go“.[189]

[187] Vgl. [Frau 2005]
[188] Vgl. [Tamm 2004] S. 4 bis 5
[189] Vgl. [Herr 2004]

Anhang

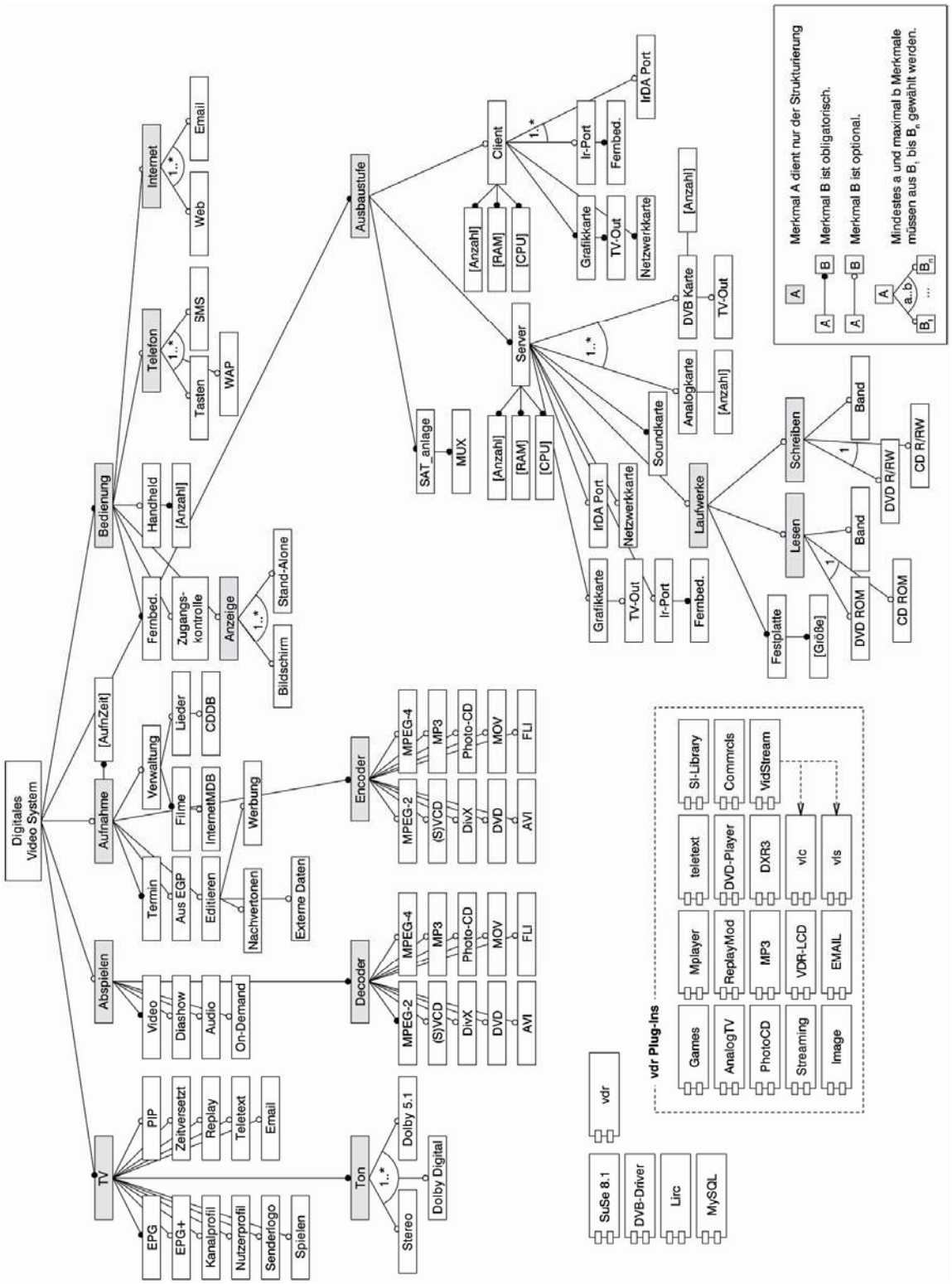
Anhang A: Aufbau des DVP-Projektes an der TU Ilmenau



Relevante Hardwarezusammensetzung:

PC mit Netzwerkkarte und IrDA-Schnittstelle
CPU: Intel Pentium 4 1500 MHz
Festplatte: 40 GB von Maxtor
Grafikkarte: ATI Technologie Inc. Radeon RV 100
DVB-Karten: Technotrend DVB Card „Budget S" und DVB Card „Premium"

Anhang B: Merkmalmodell des DVP nach FORE



Anhang C: Plugins für VDR (Stand 01.07.2005)[190]

Da einige Plugins nicht mehr gepflegt werden, konnte keine Version und kein Datum ermittelt werden.

Name	Beschreibung	Homepage	Version	Datum
(S)VCD Plugin	Wiedergeben von (S)VCDs mit dem VDR	http://www.heiligenmann.de/download/	Ver: 0.0.6c	29.11.2003
AC3 bitstreamout Plugin	AC3 bitstreamout plugin	http://bitstreamout.sourceforge.net/	Ver: test_av	14.08.2004
Activy300-LCD	LCD-Plugin für Activy300	http://home.primusnetz.de/mgeisler/alcd/	Ver: 0.0.21	22.11.2004
Actuator	Satellitenschüssel Rotor	http://club.telepolis.com/l.o/vdr/	Ver: 0.0.5	22.11.2004
admin	Administrative Tasks	http://www.htpc-forum.de/index.php?url=downloads.php	Ver: 0.3.0	02.03.2005
adzap	Werbezapper	http://www.vdr-portal.de/board/thread.php?threadid=31410&sid=	Ver: 0.0.1	13.03.2005
aide	OSD Hilfe System	http://vdr.bluox.org/download/?path=vdr-aide/	Ver: 0.0.2	09.09.2004
alcd	Activy300 LCD Display	http://home.primusnetz.de/mgeisler/alcd/	Ver: 0.0.20	09.11.2003
analogradio	Analoge Radio-Karte als Eingabegerät	http://tankwar.de/analogradio.php	Ver: 0.1.2	12.03.2005
AnalogTV Plugin	Plugin für die Ausgabe von analogen Quellen über VDR	http://akool.bei.t-online.de/vdr/analogtv/index.html	Ver: 0.9.36	22.11.2004
asterisk	Client für Asterisk voicebox	http://www.htpc-tech.de/voip/ast_plugin.htm	Ver: 0.0.2	
AudioCD	Abspielen von Audio-CDs	http://mail.pad.zuken.de/~alex/vdr/	Ver: 0.0.6	14.08.2004
Autotimer Edit	OSD Autotimer	http://www.fast-info.de/vdr/autotimeredit	Ver: 0.1.6	
BEEP	Akustische Meldung über PC Lautsprecher	http://www.deltab.de/vdr/beep.html	Ver: 0.0.2	14.08.2004
bitstreamout	Dolby Digital Ausgabe über eine Soundkarte (über Alsa)	http://bitstreamout.sourceforge.net/	Ver: 0.83a	12.04.2005
btrcu	Bluetooth Mobiltelefon als Fernbedienung	http://www.k13zoo.de/vdr/	Ver: 0.0.1	31.12.2004
burn	Brennt mehrere Filme auf eine DVD	http://www.xeatre.de/community/burn/	Ver: 0.0.5	05.12.2004

[190] Diese Tabelle beinhaltet alle Plugins der Seiten [Schm 2005] Plugins, [VDR-W 2005] Plugins und [VDR-P 2005] Downloads

Name	Beschreibung	Homepage	Version	Datum
calc	Kleiner Taschenrechner	http://www.vdrcalc.bmschneider.de/index2.html	Ver: 0[1].0.1-rc5	
Calendar Plugin	Zeigt EPG Daten in Form eines Kalenders	http://ricomp.de/vdr/	Ver: 0.1.4	02.03.2004
Camcorder	Anschließen eines Camcorders	http://home.vr-web.de/~erich.bachl	Ver: 0.0.2a	22.12.2003
channelscan	Scannt Satellitentransponder und erzeugt und aktualisiert die channels.conf.	http://kikko77.altervista.org/	Ver: 0.0.4	16.06.2005
channels-witcher	Umschalten zwischen mehreren Kanaleinstellungen	http://www.freewebs.com/sadhome/	Ver:	
Chanorg	Kanallisten Organizer	http://www.freewebs.com/sadhome/	Ver: 0.0.5	22.11.2004
cinebars	Erzeugt virtuelle Kinostreifen	http://vdrportal.de/board/thread.php?threadid=24235	Ver: 0.0.1	31.10.2004
Client/Server Streaming	Wiedergabe eines Videostreams in einem Netzwerk	http://www.magoa.net/linux	Ver: 0.0.1	11.02.2003
Clock Plugin	Uhrzeit per OSD	http://vdr.humpen.at	Ver: 0.0.5b1	14.08.2004
Commander Plugin	Dateiverwaltung im Norton-Commander-Stil über das OSD	http://www.vdr-portal.de	Ver: 0.0.8	23.02.2003
Console Plugin	Linux-Console per OSD	http://ricomp.de/vdr	Ver: 0.5.1	23.02.2003
Control	Ausgabe des OSD auf einem Terminal	http://ricomp.de/vdr/	Ver: 0.0.2a	14.08.2004
csf	Sortiert Kanäle	http://jmorra.tripod.com/	Ver: 0.01	
decruft	Löscht Channels nach Vorgaben	http://www.rst38.org.uk/vdr/	Ver:	
Digicam	Anzeige und Übertragung von Fotos	http://turku.wi-bw.tfh-wildau.de/~pjuszack/digicam	Ver: 1.0.0	01.02.2005
director	Multifeedkanäle von Premiere nutzen	http://www.wontorra.net/staticpages/index.php?page=director	Ver: 0.2.2	04.06.2005
dsmcc	Mpeg2-Decoder	http://sourceforge.net/projects/libdsmcc	Ver: 0.4.1	23.10.2003
dummydevice	Dummy-Output-Device	http://users.tkk.fi/~phintuka/vdr/	Ver:	13.06.2005
DV	DV-Camcorder-Plugin	http://home.vr-web.de/~erich.bachl	Ver: 0.0.2a	14.08.2004
dvd	DVD Spieler Plugin	http://jausoft.com/Files/vdr/vdr-dvd/	Ver: 0.3.6	24.01.2005
dvdconvert	Konvertiert DVDs	http://home.lausitz.net/lini/vdr/	Ver: 0.2.5	27.02.2005
DVDSelect	dvdselect ist ein kleines Plugin um nicht ständig DVDs wechseln zu müssen	http://vdr.sjur.de/	Ver: 0.0.7a	01.08.2004
DVB Subtitles	DVB Subtitles Plugin for VDR	http://virtanen.org/vdr/subtitles	Ver: Version 0.3.7	15.02.2005
DXR3	Nutzen der DXR3 zur Ausgabe	http://www.schluenss.de/DXR3.html	Ver: 0.2.2	14.08.2004

Name	Beschreibung	Homepage	Version	Datum
Email Reader / VDRmail	Emails lesen mit dem VDR	http://www.dapeace.de/	Ver: 0.0.2	14.08.2004
epgsearch	EPG-Suche	http://people.freenet.de/cwieninger/html/vdr-epg-search.htm	Ver: 0.0.7	07.02.2005
extb	Plugin für das VDR Extension Board	http://www.deltab.de/vdr/extb.html	Ver: 0.2.8	14.08.2004
femon	Anzeige der Signalstärke und -qualität	http://www.saunalahti.fi/~rahrenbe/vdr/femon/	Ver: 0.1.6	22.11.2004
fepg	Grafische Navigation im EPG	http://www.fepg.tk/	Ver: 0.1.3	25.07.2004
freecell	Das Spiel Freecell	http://www.magoa.net/linux/index.php?view=freecell	Ver: 0.0.2	14.08.2004
Games Plugin	Die Spiele Tetris, Tron und Snake auf dem VDR	http://1541.org/	Ver: 0.6.1	01.12.2004
gngb2vdr	GameBoy Emulator	http://vdrbox.free.fr/fichiers/	Ver:	
GraphLCD Plugin	verschiedene grafische LCDs	http://www.powarman.de/vdr_graphlcd.htm	Ver: 0.1.1	22.11.2004
GraphTFT	über eine Framebuffer-Device oder eine zweite Fullfeature-DVB-Karte Informationen ausgeben	http://www-math.upb.de/~tegeler/vdr/graphtft/	Ver: 0.0.7	22.11.2004
hello	Ein einfaches „hello“ Plugin	http://www.cadsoft.de/vdr/	Ver:	19.02.2005
Image-Plugin	Bilder mit dem VDR anzeigen	http://www.deltab.de/vdr/image.html	Ver: 0.2.2a	22.11.2004
ipod	Zeigt ITunesDB des iPod an	http://www.jwendel.de/	Ver: 0.0.1	25.12.2004
isdnlog	ISDN Log: Zeigt Telefonanrufe an	http://akool.bei.t-online.de/vdr	Ver:	
joystick-plugin	Joystick oder Gamepad als Bedienmöglichkeit	http://www.powarman.de/vdr_plugins.htm	Ver: 0.0.3	20.12.2003
kathreinlcd	I²C Displays	http://www.magoa.net/linux/index.php?view=kathrein	Ver: 0.0.1	14.08.2004
launcher	Startet andere Plugins	http://people.freenet.de/cwieninger/index.html	Ver: 0.0.2a	25.06.2005
LCD Plugin	Ausgabe von Programminformationen an ein Matrix-LCD	http://home.pages.at/linux/dvb.html	Ver: 0.0.10	21.03.2004
lcdproc	Alphanummerische Displays	http://home.pages.at/linux/dvb.html	Ver:	17.01.2004
lirc	Lirc Test plugin	http://www.wontorra.net/filemgmt/	Ver: 0.1.0	14.01.2004
loadepg	Import von EPG Daten (Canal+ group)	http://kikko77.altervista.org/	Ver: 0.1.4	16.06.2005
locker	Kinderschutz-Sendungen sperren	http://www.freewebs.com/sadhome/	Ver:	

Name	Beschreibung	Homepage	Version	Datum
mailbox	Einfacher eMail-Client	http://sites.inka.de/~W1222/vdr/	Ver: 0.3.0	14.08.2004
manual	Einlesen von XML-Daten	http://www.linvdr.org/download/vdr-manual/	Ver: 0.0.3	30.04.2005
Mediadetection Plugin	Erkennt eingelegte CDs und startet das passende Plugin	http://www.magoa.net/linux/index.php?view=vdrcd	Ver: 0.0.10	14.08.2004
MediaMVP	Anzeige von Mediendaten auf dem Fernsehbildschirm	http://www.rst38.org.uk/mediamvp/	Ver: 0.1.5	22.11.2004
message	Schnittstelle für Shell-Skripte	http://www.jwendel.de/	Ver: 0.0.1	25.12.2004
Message Viewer	Plugin, um die syslog (/var/log/messages) im VDR anzuzeigen	http://www.vdrportal.de/board/thread.php?threadid=15292	Ver: 0.01	06.08.2003
MHP-Plugin	Plugin zum Empfang von MHP-Daten	http://www.powarman.de/vdr_plugins.htm	Ver: 0.0.3	29.12.2003
MLCD	Siemens Multitainer LC-Display	http://home.arcor.de/meinrad	Ver: 0.0.4	22.11.2004
mldkgui	EDonkey Client	http://mldkgui.sourceforge.net/	Ver: 0.0.1-pre3	04.01.2005
mldonkey	Anzeige des Download-Fortschritts	http://www.federationhq.de/	Ver: 0.0.4a	23.11.2004
MP3oss Plugin	MP3s-Player über eine andere Soundkarte	http://home.pages.at/garvdr/	Ver: 0.8.0	03.07.2003
MPlayer / MP3 Plugin	Wiedergeben von MP3s und Videos (z.B. AVI) mit dem VDR	http://www.muempf.de	Ver: 0.9.10	07.02.2005
MPlayer Cluster Plugin	MPlayer im Verbund	http://www.magoa.net/linux	Ver: 0.0.1a	03.07.2003
Muggle	MP3 Spieler mit Datenbank	http.//www.htpc-tech.de/htpc/muggle.htm	Ver: 0.1.7-BETA	
Multitainer LCD	Ermöglicht die Anzeige des Siemens MULTITAINER mit dem VDR	http://home.arcor.de/meinrad	Ver: 0.0.4a	09.01.2005
network	Netzwerkeinstellungen ändern	http://kron.homeftp.net/~sebastian/	Ver:	16.06.2005
Newsticker Plugin	Darstellung eines Newstickers am Fernsehbildschirm	http://www.wontorra.net/staticpages/index.php?page=newsticker	Ver: 0.0.3	29.11.2003
osddemo	Demonstriert das On-Screen-Display	http://www.cadsoft.de/vdr/	Ver:	19.02.2005
osdimage	Ausgabe von Bildern über das OSD	http://home.pages.at/brougs78/	Ver: 03	09.06.2005
OSD Picture-in-Picture	Picture-in-Picture auf dem OSD	http.//www.magoa.net/Linux	Ver: 0.0.8	15.05.2005

Name	Beschreibung	Homepage	Version	Datum
OSD-Teletext	Bringt den Videotext aufs OSD	http://www.wiesweg-online.de/linux/vdr	Ver: 0.4.1	01.12.2004
OSD-Test 256	Plugin, um auf 4 MB gemodde-te DVB-Karten zu testen	http://endriss.escape.bei.t-online.de/vdr/vdr-osdtest256-0.1.0.tgz	Ver: 0.1.0	22.11.2004
PhotoCD	Darstellung von PhotoCD	http://vdr.heiligenmann.de/download/	Ver: 0.0.2	19.05.2003
Pilot Plugin	EPG co-Pilot für den VDR	http://famillejacques.free.fr/vdr/	Ver: 0.0.6	15.06.2004
PIM	Personal Information Manager, Darstellung eines Kalenders	http://tuffi.privat.t-online.de/vdr/	Ver: 0.0.2	22.11.2004
playlist	Verwendung von Playlisten für Videoaufnahmen	http://www.fast-info.de/vdr/playlist/index.htm	Ver: 0.0.2rc3	23.09.2004
Powermate	Bedienung des VDR über einen Joystick	http.//www.powarman.de/vdr_plugins.htm	Ver: 0.0.3	09.12.2003
Prefermenu Plugin	Kurzwahl-Menü für 10 Sender	http://www.olivierjacques.com/vdr/prefermenu	Ver: 0.6.2	14.08.2004
pvr350	Analoge TV-Karte als MPEG-Ausgabegerät (Hauppauge PVR 350)	http://www.rst38.org.uk/vdr/	Ver:	
radio	Hintergrundbild für Radiosen-der	http://homepages.uni-paderborn.de/tegeler/vdr/	Ver:	14.11.2004
Remote Plugin	Nutzen des IR-Original-Empfängers der Nexus 2.1	http://endriss.escape.bei.t-online.de/vdr/	Ver: 0.3.2	22.11.2004
Rotor	Plugin für Diseqc-Motoren	http://home.vr-web.de/~bergwinkl.thomas/	Ver: 0.0.6c	22.11.2004
sc	Software CAM (Conditional Access Modul)	http://www.openssl.org/	Ver: 0.9.8	13.06.2005
Scanner Plugin	Plugin zum Transponderscan	http://www.akool.de	Ver: 0.0.1	17.12.2003
Screenshot	Zum Erstellen von Screenshots	http://www.stud.uni-karlsruhe.de/~upi9/vdr/	Ver: 0.0.5	14.08.2004
Serial-Plugin	Plugin zur Steuerung des VDR über die serielle Schnittstelle	http://www.lf-klueber.de/vdr.htm	Ver: 0.0.6	14.08.2004
setup	Plugin zum Bearbeiten von Systemeinstellungen / VDR-Menüs usw. per OSD	http://www.vdrtools.de/	Ver: 0.05	26.02.2005
skincurses	VDR-Skin für Konsole	http://www.cadsoft.de/vdr/	Ver:	19.02.2005
skinelchi	VDR-Skin (text2skin nicht nötig)	http://vdrportal.de/board/thread.php?threadid=30834	Ver: 0.0.1	02.03.2005
skinnotrans	Eine nicht-transparente Skin	http://nothome.com/vdr/	Ver:	
sked	Ein OSD Editor	http://vdr.bluox.org/download/?path=vdr-sked/	Ver:	07.06.2005
sky x	Mpeg2 Encoder (auch als „dummy“ input)	http://www.cadsoft.de/vdr/	Ver:	19.02.2005

Name	Beschreibung	Homepage	Version	Datum
Sleeptimer Plugin	Fährt den VDR nach der angegebenen Zeit runter	http://linvdr.org/download/vdr-sleeptimer/	Ver: 0.0.6	01.08.2004
snapshot	Erstellt Screenshots	http://users.tkk.fi/~phintuka/vdr/	Ver:	13.06.2005
softdevice2net	Netzwerk-Ausgabegerät statt Framebuffer	http://nano.gmxhome.de	Ver:	
Softdevice Plugin	Framebuffer als Ausgabegerät	http://softdevice.berlios.de/	Ver: 0.0.8	06.02.2005
solitaire	Solitaire Kartenspiel	http://www.djdagobert.com/vdr/solitaire/index.html	Ver: 0.0.2	21.03.2005
status	OSD Status	http://www.cadsoft.de/vdr/	Ver:	19.02.2005
statusandquestion	Erweiterung des Svdrp Protokolls	http://www.fast-info.de/vdr/	Ver: 0.0.2	
StatusLED	Lässt eine StatusLED der Tastatur blinken	http://sourceforge.net/projects/vdr-statusleds	Ver: 0.0.7	14.08.2004
streamplayer	Video Stream Player	http://urichter.cjb.net/vdr	Ver: 0.0.1	06.02.2005
Stream Plugin	Erzeugt einen Video-Stream, der auf anderen Rechnern empfangen werden kann	http://www.magoa.net/linux	Ver: 0.1.1	18.07.2003
Streamdev Plugin	Wiedergeben eines Videostreams auf dem VDR	http://www.magoa.net/linux/index.php?view=streamdev-unstable	Ver: 0.3.1	14.08.2004
submenu	Erstellen von Untergruppen im Menü	http://www.freewebs.com/sadhome/	Ver:	
Subtitles Plugin	DVB Untertitel-Decoder	http://www.saunalahti.fi/~opvirtan/vdr/subtitles/	Ver: 0.3.3	14.08.2004
suspendoutput	Hintergrundbild für Radiosender	http://users.tkk.fi/~phintuka/vdr/	Ver:	13.06.2005
sysinfo	Systeminformationen via OSD	http://kikko77.altervista.org/	Ver: 0.0.4	16.06.2005
Taste	Erstellung einer Favoritenliste	http.//www.magoa.net/linux	Ver: 0.0.2d	09.01.2005
Teletext Plugin	Anzeige von Videotext-Seiten auf dem Fernseher	http://ffmpeg.sourceforge.net/	Ver: 0.7.6	18.07.2003
Teletext Subtitle Plugin	Zeigt Teletext Untertitel via OSD	ftp://ftp.nada.kth.se/pub/home/ragge/vdr/	Ver: 0.0.5	14.08.2004
text2skin	Skin-Plugin für VDR-Versionen höher 1.3.11	http://www.magoa.net/linux	Ver: 0.0.8.1	22.11.2004
Text-Browser-Plugin	Kleiner Browser mit Lesezeichen-Funktion	http://www.schaeben.info/vdr/	Ver: 0.1	15.08.2003
TimeLine	Timeline zeigt eine Zeitleiste aller Aufnahmen an einem Tag	http://www.js-home.org/vdr/download.php	Ver: 0.8.1	14.10.2003

Name	Beschreibung	Homepage	Version	Datum
Transfron Plugin	Umwandeln von Aufnahmen	http://www.peess.de/projects/transfron/transfron_plugin-eng.html	Ver: 0.3.2	04.12.2003
ttxtsubs	Teletext Untertitel	http://vdrportal.de/board/thread.php?threadid=17881	Ver: 0.0.5	01.06.2004
TVonScreen	Zeigt die EPG Daten in der für TV-Zeitungen üblichen Anordnung	http://www.js-home.org/vdr/tvonscreen/index.php	Ver: 0.6.0	22.11.2004
TVTV Plugin	Programmieren von Aufnahmen über www.tvtv.de	http://www.vdr-portal.de	Ver: 0.1.6	23.02.2003
Undelete	Wiederherstellung gelöschter Aufnahmen	http://www.fast-info.de/vdr/undelete	Ver: 0.0.2	
usbremote	Remote Plugin für Plug-USB Fernbedienungen (AVR)	http://homepages.uni-paderborn.de/tegeler/vdr/	Ver: 0.0.1	14.11.2004
Vbox Plugin	Oberfläche für den VBox Anrufbeantworter	http://linvdr.org/download/vdr-vbox/	Ver: 0.5.1	21.08.2004
vdr-burn Plugin	Plugin zum Erstellen von DVDs aus VDR-Aufnahmen	http://www.xeatre.de/community/burn/	Ver: 0.0.4b	01.12.2004
vdrc	Commander File Manager	http://vdrportal.de/board/portal_downloads.php	Ver:	21.02.2005
vdrcd	AutoPlay Funktion	http://www.magoa.net/linux/	Ver:	
vdricq	ein ICQ-Client-Plugin	http://vdr.computer-wiki.de/	Ver: 0.0.2	17.04.2005
VDRRIP - Plugin	Umwandlung von VDR Aufnahmen in das DivX Format	http://www.a-land.de/	Ver: 0.3.0	14.05.2004
WAP-Deamon	Fernsteuerung durch WAP	http://vdr.heiligenmann.de/download/	Ver: 0.0.6d	22.11.2004
Wetter-Plugin	Das Wetter-Plugin für VDR-1.3.x	http://www.moldaner.de/vdr/	Ver: 0.2.1e	22.11.2004
X11out	X11 auf der DVB ausgeben	http://www.js-home.org/vdr/x11out/	Ver: 0.0.1	14.08.2004
Xinelibout	X11 Front-End für VDR	http://www.hut.fi/u/phintuka/vdr	Ver: 0.3	17.02.2005
XINE-Plugin	Software-Emulation eines VDR-Ausgabegerätes	http://home.vr-web.de/~rnissl	Ver: 0.7.4	08.05.2005
YaEPG	Eine etwas andere EPG-Ansicht	http://htpc.at/public/vdr/yaepg	Ver: 0.0.2	22.11.2004
zaphistory	Zapping History	http://www.vdr-portal.de/board/thread.php?threadid=31492&sid=	Ver: 0.0.1	15.03.2005

Thesen

1. Digitalfernsehen ist nur eine von vielen Möglichkeiten, das Leben multimedial zu vernetzen. Basierend auf diesem Trend verändern sich Kundenanforderungen stetig und erhöhen den Druck auf den Markt. Der Absatzbreite von Endgeräten wächst proportional.

2. Die Entwicklung von Systemfamilien gewinnt aufgrund der höher werdenden Anforderungen bezüglich Zeit, Qualität und Kosten und der kürzer werdenden Softwareentwicklungszyklen an Bedeutung, da Systemfamilien effizient Wiederverwendung nutzen.

3. Die vollständige und weitgehend fehlerfreie Analysephase mit der Erhebung der Kundenanforderungen wird immer komplexer; die Ergebnisse dieser Phase entscheiden grundlegend über den Projekterfolg. Diese Phase wird in Zukunft wesentlich ausgeprägter und intensiver erfolgen müssen, da die Fehlerkosten und der Aufwand zur Fehlerbehebung massiv steigen.

4. Die vollständige Entwicklung einer Systemfamilie und die automatische Ableitung von Familienmitgliedern sind trotz vorhandener Konzepte und Werkzeuge langwierige und schwer zu managende Prozesse; der massive Aufwand kann unter Umständen erst lange Zeit später zum Erfolg führen.

5. Ein bestehender Automatismus zur Ableitung von Familienmitgliedern erhöht die Softwarequalität in allen genannten Kriterien und optimiert so das Erfolgsdreieck Zeit, Kosten und Qualität. Auch mit anfänglich großem Mehraufwand wird die generative Softwareentwicklung die konventionelle bei großen Systemfamilienprojekten ablösen oder zumindest stark ergänzen.

6. Die flexible Struktur von Komponenten- und Pluginarchitekturen wird verstärkt für neue Softwareprojekte verwendet werden, da sie die Softwarequalität in vielen Bereichen positiv beeinflusst.

7. Gerade im Bereich Opensource werden viele innovative Projekte wie die VDR-Software oder Eclipse initiiert. Durch die breite Entwicklergemeinde nimmt die Entwicklung moderner Lösungen, Werkzeuge und Konzepte zu. Die Bedeutung von Opensource-Software und Linux wächst stetig.

Literaturverzeichnis

[Alex 2004] Mathias Riebisch et al: Alexandria Methodology Home.
www.theoinf.tu-ilmenau.de/~pld/
TU Ilmenau, 2004.
Abruf : 12.06.2005.

[Apac 2005] The Apache Software Foundation
http://httpd.apache.org/docs/misc/FAQ.html
Abruf: 23.06.2005.

[Arno 2005] Arne Arnold und Christian Mohr: Implementierung im Domain und Application Engineering - Technische Aspekte von Software-Produktlinien.
http://www.informatik.fh-mannheim.de/~knauber/MSc-MSI/B03_midQual.pdf
FH Mannheim, 2005.
Abruf: 15.07.2005.

[Balz 2001] Helmut Balzert: Lehrbuch der Software- Technik, Software- Entwicklung.
2. Auflage, 1. Nachdruck, Spektrum Akademischer Verlag GmbH, 2001.

[Bato 2005] Don Batory: Feature Oriented Programming for Product-Lines.
http://www.cs.utexas.edu/users/schwartz/BatoryTutorial.pdf
Abruf: 14.07.2005.

[Baue 2000] Günter Bauer: Bausteinbasierte Software – Eine Einführung in moderne Konzepte des Software-Engineering.
Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2000.

[Benz 2003] Michael Benz: Einführung in die Generative Programmierung.
http://esche.fzi.de/proseminare/SS2003/Ausarbeitungen/GP-Ausarbeitung.pdf
FZI Forschungszentrum Informatik, 2003.
Abruf: 18.07.2005.

[Bodn 2005] Ladislav Bodnar: DistroWatch.com: Put the fun back into computing. Use Linux, BSD.
http://distrowatch.com
Abruf 16.04.2005.

[Born 1998] Günter Born: Arbeiten mit der Registrierung von Windows 98.
Microsoft Press Deutschland, 1998.

[Brüg 2005] Hans H. Brüggemann: Generatoren für Datenbanksysteme.
http://www.dbs.uni-hannover.de/~jb/#generatoren
Universität Hannover, 2005.
Abruf: 23.05.2005.

[CT-SO 2004] C'T Sonderheft: Kino daheim.
Heise Verlag Zeitschriften, 2004.

[Czar 1998] Krzysztof Czarnecki: Generative Programming – Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models.
Dissertation an der TU Ilmenau, 1998.

[Czar 2000] Krzysztof Czarnecki und Ulrich W. Eisenecker: Generative Programming – Methods, Tools and Applications.
Addison-Wesley Verlag, 2000.

[Czar 2001] Krzysztof Czarnecki, Lutz Dominick, Ulrich W. Eisenecker: Aspektorientierte Programmierung in C++.
http://www.heise.de/ix/artikel/2001/08/143/
Heise Verlag, 2001.
Abruf: 03.07.2005.

[Czar 2001-2] Dr.-Ing. Krzysztof Czarnecki: Generative Programmierung und die Entwicklung von Softwaresystemfamilien.
http://www.tu-chemnitz.de/informatik/HomePages/Betriebssysteme/IFKOLL/2001/docs/k53.pdf
TU Chemnitz, 2001.
Abruf: 20.06.2005.

[Dig-F 2005] DIGITALFERNSEHEN.de - Die Nummer 1 zum Thema Digital TV.
http://www.digitalfernsehen.de/
Abruf: 05.04.2005.

[DIN 10007] Qualitätsmanagement - Leitfaden für Konfigurationsmanagement (ISO 10007:2003).

[DVB-O 2005] DVB – The Standard of The Digital World.
http://www.dvb.org
Abruf: 03.04.2005.

[DVBT 2005] TechniSat Digital GmbH: DVB-T - das digitale Überall-Fernsehen.
http://www.dvb-t-portal.de
Abruf: 05.04.2005.

[DVP 2005] Detlef Streitferdt: Digital Video Project.
http://proinf.de/DVP
Abruf: 08.04.2005.

[Eise 2002] Ulrich W. Eisenecker, Mathias Henss, Markus Lang, Max Schlee: Generative Programmierung.
http://www.old.netobjectdays.org/pdf/02/papers/indepth/gen_progr_text.pdf
Abruf: 19.06.2005.

[Eise 2002-2] Ulrich W. Eisenecker, Mathias Henss, Markus Lang, Max Schlee: Generative Programmierung. Foliensatz.
http://www.old.netobjectdays.org/pdf/02/slides/indepth/gen_progr_folien.pdf
Abruf: 19.6.2005.

[Eise 2003] GP-WEB Team der Fachhochschule Kaiserslautern, Projektleiter Prof. Dr. Ulrich W. Eisenecker et al: Generative Programmierung für web-orientierte Softwaresystemfamilien.
http://gp.informatik.fh-kl.de/index.php?goto=home&lang=de
FH Kaiserslautern, 2003.
Abruf: 12.06.2005.

[Eise 2003-2] Ulrich W. Eisenecker und Christof A. Hurst: Generative Programmierung für web-orientierte Softwaresystemfamilien - Ein E-/Web-Learning-Projekt. Projektbericht.
GP-WEB Team der Fachhochschule Kaiserslautern.
http://gp.informatik.fh-kl.de/downloads/doku/Projektbericht.pdf
FH Kaiserslautern, 2003.
Abruf: 12.06.2005.

[Eise 2003-3] Ulrich W. Eisenecker et al: Generative Programmierung für web-orientierte Softwaresystemfamilien. Kick-Off Meeting.
http://gp.informatik.fh-kl.de/downloads/kickoff/fh_kl_eisenecker.pdf
FH Kaiserslautern, 2003.
Abruf: 12.06.2005

[Fest 2005] Festplattenrecorder.com - Informationen, Produkte und Trends zum Thema Festplatten-Videorecorder.
http://www.festplattenrecorder.com.
Abruf: 11.03.2005.

[For-D 2005] Forumdeluxx: Your Guide for Luxurious Hardware.
http://www.forumdeluxx.de/
Abruf: 12.04.2005

[Fres 2005] Fresh ISOs, like mum used to burn: Linuxiso.org.
http://www.linuxiso.org/
Abruf: 22.04.2005

[Frau 2005] Fraunhofer Institut IESE: Experimentelles Software Engineering - Projects.
http://www.iese.fraunhofer.de/Projects/
Abruf: 20.07.2005.

[Gamm 1996] Erich Gamma et al: Entwurfsmuster – Elemente wieder verwendbarer objektorientierter Software.
1996, 1. Auflage, Addison Wesley Longman Verlag GmbH.

[GARV 2005] Gesellschaft zu Förderung der Rundfunkversorgung mbH.
http://www.garv.de
Abruf: 18.04.2005.

[GB-PV 2004] GB-PVR.
http://www.gbpvr.com/
Abruf: 12.05.2005.

[GNO-D 2005] GNOME Deutschland.
http://www.gnome-de.org/gnome-2.8.php
Abruf: 06.06.2005.

[Gruh 2004] Volker Gruhn: Ausgewählte Kapitel der Software-Technologie - Testen.
ebus.informatik.uni-leipzig.de/www/media/lehre/st-test04/st-test04-ve11-ppt.ppt
Universität Leipzig, 2004.
Abruf: 22.07.2005.

[Hand 2005] Dr. Haimo L. Handl: Handl Management Consulting.
http://www.handl.net
Abruf: 01.07.2005.

[Hatt 2005] Rainer Hattenhauer: Linux-Livesysteme – Knoppix, Ubuntu, Morphix, Kanotix & Co.
Galileo Press GmbH Bonn, 2005.

[Hei-F 2004] C't Projekte: Fernseh-PC.
http://www.heise.de/ct/ftp/projekte/vdr3/download.shtml
Abruf: 12.05.2005.

[Heim 2005] Heimkino.at: Wir bringen das Kino zu Ihnen nach Hause.
http://www.heimkino.at
Abruf: 12.04.2005.

[Heis 2005] Das Online- Format der Heise Verlag Zeitschriften.
http://www.heise.de
Abruf: 02.04.2005.

[Herr 2004] Jack D. Herrington: Code Generation Network - Krzystof Czarnecki on Generative Programming.
http://www.codegeneration.net/tiki-read_article.php?articleId=64
Abruf: 25.07.2005.

[INPR 2005] Webdesign, edv & telecommunication company: INPROT.
http://www.inprot.at
Abruf: 01.07.2005.

[Inte 2005] Interlogin Software GmbH.
http://www.interlogin.net
Abruf: 01.07.2005.

[Java 2005] Sun Developer Network: Products and Technologies.
http://java.sun.com/products/javabeans/
Abruf: 20.06.2005.

[Kab-B 2005] Kabel Baden-Württemberg GmbH & Co. KG.
http://www.kabelbw.de
Abruf 18.04.1005.

[Kab-D 2005] Kabel Deutschland: Wir bringen das digitale Fernsehen.
http://www.kabeldeutschland.de
Abruf 18.04.1005.

[Kano 2005] Kano: Have Fun with Kanotix!
http://kanotix.com
Abruf: 12.04.2005.

[Kan-W 2005] Kanotix-Wiki, der Wiki der Linux-Distribution KANOTIX!
http://wiki.kanotix.net
Abruf 16.04.2005.

[KDE 2005] K Desktop Environment - Conquer your Desktop!
http://www.kde.org/
Abruf: 31.05.2005.

[Knop 2005] Klaus Knopper: Live Linux Filesystem on CD.
http://www.knopper.net/knoppix/
Abruf: 15.03.2005.

[Koch 2004] Thomas Koch und Mirko Dölle: LinVDR News.
http://linvdr.org/projects/
Abruf: 23.04.2005.

[Kofl 2003] Michael Kofler: Linux – Installation, Konfiguration, Anwendung.
Addison-Wesley Verlag, 2003.

[Kuhl 2004] Urs Kuhlmann, Andreas Winter und Harry M. Sneed: Workshop Reengineering Prozesse.
http://www.uni-koblenz.de/~ist/repro2004/aufruf.html
Universität Koblenz-Landau und Universität Regensburg, 2004.
Abruf: 20.07.2005.

[Lage 2004] Bart Lagerweij: Bart's Preinstalled Environment (BartPE) ein von CD/DVD bootbares Live Windows.
http://pcfreaks.big-clan.net/bartpe/pebuilder.shtml
Abruf: 12.05.2005.

[Lani 2000] Ivan van Laningham:
Jetzt lerne ich Python – Der schnelle Einstieg in die wunderbare Welt der Programmierung.
Markt+Technik Verlag, 2000.

[Lemp 2002] Sebastian Lempert, Hauke Traulsen: Enterprise JavaBeans™ 2.0.
http://www.inf.fu-berlin.de/lehre/SS02/KBSE/doc/ejb2/Enterprise%20JavaBeans%202.0.pdf
Institut für Informatik, Freie Universität Berlin, 2002.
Abruf: 12.06.2005.

[LiFo 2005] Linuxforen.de – Das deutschsprachige Linuxforum – User helfen Usern.
http://www.linuxforen.de/forums/showthread.php?t=117571
Abruf: 08.07.2005.

[Lin-G 2005] Linux Gazette | Making Linux just a little more fun! http://www.linuxgazette.com
Abruf: 23.04.2005

[Lin-M 2005] Linux-Magazin – Die Zeitschrift für Linux-Professionals..
http://www.linux-magazin.de
Abruf: 14.06.2005.

[Lin-T 2005] LinuxTag 2005.
http://www.linuxtag.org
Abruf 06.05.2005.

[Lin-U 2003] LinuxUser - Das Magazin für die Praxis - LinuxUser 06/2002: GNOME 2.
http://www.linux-user.de/ausgabe/2002/06/058-gnome2/gnome2.html
Abruf: 24.06.2005.

[Lin-W 2005] Thomas Waldmann et al: Willkommen auf LinuxWiki.org, einer Freiform-Text-Informationsdatenbank für alles, was mit GNU/Linux zusammenhängt.
http://www.linuxwiki.de/
Abruf: 23.04.2005.

[Live 2005] A list of all available LiveCDs and LiveDVDs.
http://www.livecdlist.com/
Abruf: 11.05.2005.

[Micr 2005] Microsoft – COM: Component Object Model Technologies.
http://www.microsoft.com/com/default.mspx
Abruf 10.06.2005.

[Micr 2005-2] Office-Anwendungen und XML
http://msdn.microsoft.com/library/deu/default.asp?url=/library/DEU/modcore/html/deconOfficeApplicationsXML.asp
Abruf: 02.07.2005.

[Mueh 2005] LinVDR auf einem 128MB USB Stick.
http://vdr.muehlbradt.com
Abruf: 23.05.2005.

[OOo 2005] de.OpenOffice.org - die deutschsprachigen Seiten.
http://de.openoffice.org/
Abruf: 13.05.2005.

[OSTG 2004] Open Source Technology Group: The world's largest development and download repository of Open Source code and applications.
http://sourceforge.net/projects/pylets/
Abruf: 24.07.2005.

[PC-MA 2005] PC Magazin Online, mit News, Tests, Tips, Infos
http://www.pc-magazin.de.
Abruf: 03.05.2005.

[PCFo 2005] Das Computerforum für alle im Netz!!! – Portal.
http://www.pcforum24.de/viewtopic.php?p=1642
Abruf: 10.07.2005.

[Phil 2003] Ilka Philippow, Detlef Streitferdt, Matthias Riebisch: Design Pattern Recovery in Architectures for Supporting Product Line Development and Application.
http://www.theoinf.tu-ilmenau.de/%7Estreitdf/TheHome/own
/data/ECOOP-WS-DesPattRec.pdf
Abruf: 20.07.2005.

[Phil 2005] Ilka Philippow: Vorlesung Grundlagen der Informatik – 2. Semester.
http://www.tu-ilmenau.de/fakia/uploads/media
/Vorlesung_2_Semester.pdf
Abruf: 03.07.2005.

[Punz 1999] Werner Punz : Komponenorientierte Programmierung.
http://witiko.ifs.uni-linz.ac.at/research/masters/punz/Dipltext/index.html
Universität Linz, 1999.
Abruf: 23.07.2005.

[Pure 2005] pure-systems GmbH: pure::varinants – Variantenmanagement von Softwareprodukt-linien.
http://web.pure-systems.com/Variant_Management.49.0.html
Abruf: 22.07.2005.

[Qual 2005] Qualitätsmanagement unter einem D,A,CH.
http://www.quality.de
Abruf: 01.07.2005.

[Rich 1999] Michael Richter: Online-Befragung als neues Instrument zur Beurteilung der Benut-zerfreundlichkeit interaktiver Software am Beispiel einer Internet-Anwendung.
ISO 9241-10 (1996) Ergonomische Anforderungen für Bürotätigkeiten mit Bild-schirmgeräten - Teil 10: Grundsätze der Dialoggestaltung. Brüssel: CEN
http://www.intuitive.ch/richter/gor_99.pdf
Abruf: 12.07.2005.

[Rieb 2000] Matthias Riebisch, Detlef Streitferdt, Kai Böllert: Methoden und Werkzeuge zur Entwicklung von Systemfamilien.
http://www.theoinf.tu-ilmenau.de/~riebisch/pld-old/pub/2000/oose-slides.pdf
TU Ilmenau, 2000.
Abruf: 12.07.2005.

[Rieb 2004] Dr.-Ing. habil. Matthias Riebisch: Themen für studentische Arbeiten.
http://www.theoinf.tu-ilmenau.de/~riebisch/studarb.htm
TU Ilmenau, 2005.
Abruf: 24.06.2005.

[Robi 1998] Paul Robichaux: Windows NT Registrierung – Für Systemadministratoren.
O'Reilly Verlag, 1998.

[Rpms 2005] Rpmseek.com: Die Suchmaschine für Linux rpm und Debian Pakete.
http://www.rpmseek.com
Abruf: 18.05.2005.

[Sage 2005] SageTV: You're in control.
http://www.sage.tv
Abruf: 11.05.2005.

[Saou 2005] Matthias Saou: Welcome to freshrpms.net. Simple, clean... and rpm packages.
http://www.Freshrpms.net
Abruf: 16.05.2005.

[Schm 2005] Klaus Schmidinger: VDR Projekt.
http://www.cadsoft.de/people/kls/vdr
Abruf: 20.04.2005.

[Schö 2005] Melanie Schöppe, Martin Rulsch: Digitales Video-Projekt auf einer Boot-CD.
Studienjahresarbeit an der TU Ilmenau, 2005.

[Schw 2005] Dr. Holger Schwichtenberg: Objektorientierte Komponentenarchitekturen.
http://www.com-komponenten.de
Abruf: 19.06.2005.

[Seib 2005] Siegfried Seibert: Wegweiser. Lehrveranstaltungen im SS 2005.
http://www.siegfried-seibert.de/Wissensspeicher/PMGlossar
FH Darmstadt.
Abruf: 01.07.2005.

[Snee 2001] Harry .M Sneed: Objektorientiertes Reengineering.
http://www.caseconsult.de/fileadmin/internet/files/papers/HS_GI_OOReengineering-1-2001.pdf
Abruf: 10.07.2005.

[Stre 2004] Detlef Streitferdt: Family-Oriented Requirements Engineering. Dissertation an der TU- Ilmenau, 2004.

[Stre 2005] Detlef Streitferdt: Komponentenbasierte Softwareentwicklung.
http://www.theoinf.tu-ilmenau.de/~streitdf/KBSE/
TU Ilmenau, 2005.
Abruf: 24.06.2005.

[Stru 2005] Wolfgang Strunk: Softwarearchitektur.de.
www.softwarearchitektur.de
Abruf: 12.07.2005.

[Tamm 2004] Tammo van Lessen: Generatives Programmieren.
http://www.taval.de/pub/tl-finale.pdf
Universität Stuttgart, 2004.
Abruf: 12.07.2005.

[Ue-TV 2005] DVB-T: Das ÜberallFernsehen.
Deutsche TV-Plattform e.V. c/o ZVEI
http://www.ueberall-tv.de
Abruf: 23.04.2005.

[Unit 2005] www.unitec.it GmbH Procurement Outsourcing, supply chain management in outsourcing and Industrial brokerage.
http://www.unitec.it/de/home.php
Abruf: 01.07.2005.

[VDR-P 2005] Das VDR Portal.
http://www.vdr-portal.de
Abruf: 24.04.2005.

[VDR-W 2005] Das Wiki Nachschlagewerk zum Video Disk Recorder (VDR) von Klaus Schmidinger.
http://www.vdr-wiki.de
Abruf: 24.04.2005.

[Völt 2000] Markus Völter: Frameworks und Komponenten - Widerspruch oder Ergänzung.
http://www.voelter.de/data/articles/FrameworksAndComponents.pdf
Abruf: 18.07.2005.

[Völt 2004] Markus Völter, Klaus Marquardt: Plug-Ins – Applikationsspezifische Komponenten.
http://www.voelter.de/data/articles/PlugInComponents.pdf
Abruf: 18.07.2005.

[Vran 2002] Zdenko Denny Vrandecic: Entwurfsmuster (Design Patterns)
http://www.nodix.de/download/entwurfsmuster.pdf
Seminararbeit an der Univerität Stuttgart, 2002.

[Webe 2005] Peter Weber: Fernsehen am Laptop
http://mitglied.lycos.de/peterweber69/
Abruf: 10.05.2005.

[Weis 1999] Dr. Anette Weisbecker, Jörg Kunsmann, Alexander Ulrich, Erwin Schuster: Komponentenbasierte Software-Entwicklung für Produkte und Dienstleistungen. Fraunhofer IAO.
Information Management & Consulting, Nr. 2/99, Mai 1999, S. 19-23

[Wiki 2005] Wikipedia: Die freie Enzyklopädie.
http://de.wikipedia.org
Abruf: 03.04.2005.

[Xfce 2005] Olivier Fourdan: Xfce Desktopumgebung.
http://www.xfce.org/
Abruf: 17.05.2005.

[Xfld 2005] os-cillation: Xfld - Xfce live demo.
http://www.xfld.org
Abruf : 20.07.2005.

[Xian 2005] xiando Corp.: LinuxReviews - Linux News and Information.
http://linuxreviews.org
Abruf: 12.05.2005.

[zpe-g 2004] Welcome to the glossary for innovation and digital product.
http://www.zpe-glossar.ethz.ch/index.php
Abruf: 01.07.2005.

Erklärung

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen angefertigt. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Quellen entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ilmenau, den 01. August 2005 Katharina Berg