Berater-Profil – Langversion

Profession: Informationstechnische Unternehmensberatung
Software- und IT-System-Architekt
Opensource-Experte
Linux Kernel Maintainer
Embedded-Linux Experte
DevOps-Experte
IT-Veteran

Kontakt: metux IT consult, c/o Enrico Weigelt
Anschrift: Fachgraben 2a, DE-98693 Ilmenau
Mobil: +49-151-27565287
eMail: info@metux.net

Letzte Änderung: 2. September 2019

1 Skills

1.1 Spezialgebiete

- GNU/Linux-Experte *(from scratch und alle gängigen Distros)*
- Linux-Embedded *(Kernel, BSP, HIL, IOT, Realtime, ...)*
- Sofware Development Management and Release Engineering
- Source Control Management
- Anwendungs-/System-Integration, Datensynchronisation, zentrales Provisioning, etc
- Cloud-Applications
- Embedded-Linux / Small-Devices
- Plattformübergreifende Portierung
- Code-Audits, Cleanups, Build-Management, Refactoring
- Laufzeit-Migration
- Paketierung, Distribution, Configuration-Management
- Protokoll- und Datenanalyse, Forensik
- Verteilte Systeme / Grid / Cloud / IOT

1.2 Branchenerfahrung

- Automotive *(zB. Bahntechnik, Tankfahrzeuge, ...)*
- Medizinprodukte / Medizintechnik *(zB. IEC-62304)*
- Finanzwesen - internationale Banken, Börsenhandel, Payment/Clearing
- Nachrichtendienste
- IT Geräteentwicklung *(Embedded systems, IOT)*
- Internetprovider + Telco *(selbst ISP/Operator)*
- Medizin *(Ärzteverbände)*
- Maschinenbau / Anlagenbau / Gebäudetechnik
- Öffentliche Verwaltung und Rechtspflege

- Politik und NGOs *(selbst Abgeordneter)*
- Meßtechnik und -Kalibrierung
- Autohäuser und KFZ-Werkstätten
- Sport- und Kultur-Verbände
- Werbebranche
- Gastronomie
- Einzelhandel
- Textil-Manufaktur
- Neue Medien

1.3 Softwaremethoden / -Gebiete

- OO-Methoden und -Programmierung *(components, signals, etc)*
- Service-orientierte Architekturen
- Funktionale Programmierung
- Logikorientierte Programmierung
- Constraint based programming + Unit-Tests
- Hardwarenahe Programmierung
- Embedded Systems
- Netzwerkprogrammierung, Protokollentwicklung und -Analyse
- Datenmodellierung *(logisch, workflow, physisch)*
- Verteilte Systeme / Grid-Computing / Cloud-Computing / IOT

1.4 Programmiersprachen

- Pascal *(15 Jahre)*
- BASIC *(18 Jahre)*
- C/C++ *(über 20 Jahre)*
- Unix-Shellprogrammierung *(20 Jahre)*
- Oberon/Modula *(9 Jahre)*
- postgreSQL PL/SQL *(15 Jahre)*
- Perl *(13 Jahre)*

- PHP *(13 Jahre)*
- Java *(15 Jahre)*
- HTML + CSS *(15 Jahren)*
- LaTeX *(seit 15 Jahren)*
- Javascript/ECMA-Script *(15 Jahre)*
- Lua
- Prolog
- Assembler
- XUL - XML Userinterface Language
- Python
- Ruby / Rails
- Haskell
- m4
- tcl/tk

1.5 Programmierwerkzeuge, Toolkits, Libraries

- *nix utils *(awk,sed,perl,...)*
- **Builders:**
  GNU make, Imake, Autoconf/Autotools, Unitool, TreeBuild
- **SCM:**
  Git, CVS, Perforce P4, Subversion, Redmine, Bugzilla, Trac
- GNU Toolchains
- **GUI-Toolkits:**
  Gtk, Qt, Motif, Xt, Xaw
- **Code-Generators:**
  Yacc, Lex, Bison
- **Database tools and libraries:**
  unixODBC, JDBC, libpq *(PostgreSQL)*, mySQL, sqLite, BerkeleyDB
- **Security:**
  OpenSSL, GNU TLS, OpenSSH
- **XML:**
  SAX, Expat
- **Graphics:**
  X11, SDL, DirectFB, Linux-FB, Wayland

- **Networking/IPC:**
  SysV-IPC, 9P2000, Plan9Port *(Plan9 on Unix)*, NetBEUI, TCPIP
- **Typesetting:**
  LaTeX, Docbook
- **IDE:**
  Emacs, Borland, Eclipse, MSVS, KDevelop

1.6 Betriebssysteme

- GNU/Linux - Embedded, from Scratch und div. Distros
- xBSD
- andere UNIXe
- Plan-9
- Native Oberon
- MS-DOS
- Novell Netware
- Windows 3.1
- Windows 9x
- Windows NT/2000/XP
- ReactOS
- L4
- GecOS
- Android

1.7 Datenbanken

- postgreSQL *(seit 15 Jahre Projekterfahrung)*
- mySQL *(seit 15 Jahren Projekterfahrung)*
- sqLite
- LDAP
- Berkeley-DB + nDBM + GNU DBM
- Xindice
- Interbase/Firebird

- Sybase
- ZopeDB
- Synthetische Filesysteme

1.8 Enterprise- und Branchen-Produkte

- Zimbra Collaboration Suite *(Core customizing, Extensions, Packaging, Operating)*
- Plone Portal Platform *(Core customization, Extensions, Packaging, Operating)*
- OpenERP *(Core customization, Extensions, Packaging, Operating)*
- Crashplan Enterprise Backup *(Packaging, Operating)*
- Proxmox VE Virtualization Platform *(Customization, Packaging, Operating)*
- RSA Security Appliance
- Elektronisches Gerichts- und Verwaltungspostfach *(EGVP/OSCI)*

1.9 Softwarepakete (incl. Entwicklung)

- Paketmanagement-Technologien *(zmpkg, dpkg/apt, rpm/yum, ipkg, etc)*
- Apache Webserver *(Entwickler des Multiplexer MPM)*
- Nginx Webserver
- Lighttpd Webserver
- Android
- Chromium
- Qt5
- Linux-Kernel
- sendmail Mailserver *(incl. selbstentwickelte Admin-Tools)*
- qMail *(Entwicklung/Paketierung)*
- X-Window, GNOME, KDE, gnuStep *(GUI-Programmierung)*
- GCC *(GNU compiler collection)* und andere GNU Tools
- Mozilla + XUL *(Admin, Entwickler, Crosscompiling)*
- OpenOffice *(Admin, Entwickler)*
- Ghostscript
- CRM: vTiger *(Entwicklung)*

- CMS: Joomla, Content*Builder, Typo3, Plone
- OpenERP
- Virtualisierung: VMware, QEmu, VirtualBox, DosBOX, WinE, KVM, XEN, OpenVZ, lGuest, LXC
- Plan9 from outer space
- zlib *(Contributor)*
- Midnight Commander *(Entwickler)*
- Zimbra *(Entwickler, Portierung, Customization)*
- u.v.m.

1.10 Hardware

- **Architekturen/Plattformen:**
  intel/x86, x86_64, m68k, ppc, arm, Commodore, Embedded *(zB. Router, Bahntechnik, Maschinenbau)*
- **Interfaces:**
  RS282, RS485, CAN, ISA, USB, ATA/IDE, SCSI, WLAN, Ethernet, DSL, Cable, ISDN, SDH, SPI, SSI, MVB, ...

1.11 Netzwerke

- **Transport-Protokolle:**
  TCP/IP *(IPv6, IPv6, IPSEC, IPIP, route management)*, IPX, NetBEUI, PPP, SLIP
- **Anwendungs-Protokolle:**
  SSL, HTTP, SOAP, XMLRPC, JSONRPC, SMTP, UUCP, Streaming, SSH
- **Management-Protokolle:**
  SNMP, RADIUS, HTCPCP, BGP
- **Netzwerk-Filesysteme:**
  NFS, 9P2000, SMB, Coda
- **Server-Anwendungen:**
  Apache HTTPD, Tomcat, Jetty, NGinx, *Zimbra Collaboration Suite* , Sendmail, QMail, ISC DHCP, NPFS

1.12 Web-Technologien

- HTML, CSS, XML, Templates

- XUL
- AJAX, Meshups, XMLRPC, SOAP, JSONRPC
- Java Applets
- Java Servlets / JSP
- Native Java
- Scanner / Crawler, Suchmaschinen
- Web-Services
- REST
- raw HTTP

1.13 Beteiligung an Opensource-Projekten (kleine Auswahl)

- ProxmoxVE Enterprise Virtualisation Platform: Packaging, Bugfixing
- Zimbra Collaboration Suite: Entwicklung, Core Customizing, Packaging, eigene Produkte
- Plone Portal Platform
- Redmine ticketing / project management system
- Git SCM
- Ubuntu GNU/Linux Distribution - 3rd-party packaging
- Joomla CMS + Extensions: Entwicklung, QM
- Apache Webserver - multiplexer MPM: Initiator/Maintainer
- Expat XML Parser: Cleanups, Crosscompiling
- esound *(Enlightenment sound server)*: Crosscompiling / Build fixes
- libpcre *(GNU regex library)*: Crosscompile fixes
- GNU readline library: Crosscompile / build fixes
- tdb *(standalone fork aus Samba)*: Entwickler/Maintainer
- procmail mail filter: Build fixes und neue Features
- BusyBox: Cleanups, Crosscompiling
- zLib Compression Library: Entwickler, Buildsystem + Crosscompile
- bZip2 Compression Tool + Library
- Bash *(bourne again shell)*: Crosscompile/Build fixes
- libaudiofile: Crosscompiling und Bugfixes
- GNU coreutils: Crosscompile und Buildfixes

- Mozilla Suite: Buildsystem, Cleanups
- net-tools: Crosscompile fixes + Cleanups
- mii-tool *(standalone-fork aus net-tools)*: Entwickler/Maintainer
- Crosstool *(crosscompiling suite)*
- OSS QM Taskforce: Initiator/Maintainer
- Comprehensive Source Database: Initiator/Maintainer
- OpenVZ *(Linux virtualization)*: Fixes, Crosscompiling
- Linux-Vserver *(Linux virtualization)*: Fixes, Crosscompiling
- coLinux *(Linux virtualization)*: Fixes
- GPM Console mouse server and library
- OpenSSL: crosscompilation + cleanups
- Plan9 / plan9port *(Unix-Portierung)*: Build fixes, standalone packages
- PkgConfig: crosscompile support
- CODA distributed filesystem
- PHP engine: crosscompilation + cleanups
- Linux VLAN: crosscompilation + cleanups
- Xwindow/X11/Xorg: modular buildsystem cleanups + crosscompilation
- BIND *(Nameserver)*: build cleanups + crosscompilation
- iproute2: build cleanups + crosscompilation
- patTemplate *(PHP template engine)*: Entwickler
- Links web-browser: buildsystem + crosscompilation
- Midnight Commander: buildsystem + crosscompilation, virtual filesystem *(MVFS)*, core developer
- MII-Tool standalone package: maintainer + buildsystem
- ISC DHCPd - modular branch
- lSSTV *(sstv protocol package)*: maintainer
- Spamassassin: standalone spamc package
- SU-wrapper: Entwickler/Maintainer
- Xmeter: Entwickler/Maintainer
- CryoSleep: Entwickler
- Unitool *(universal toolchain frontend)*: Entwickler/Maintainer
- TreeBuild *(buildsystem w/ declarative model)*: Entwickler/Maintainer

- Linux-Kernel: crosscompiling, 9P-fs, Coda-FS, div. Treiber
- Libreoffice
- Plan9
- Nebulon Storage Grid: Initiator/Maintainer
- Android *(Kernel + Userland)*
- Chromium
- FreeCol
- PtxDist
- u.v.m

2 Referenzprojekte

Im Folgenden Abschnitt wird eine Auswahl realisierter Projekte *(unsortiert)* vorgestellt.

2.1 MySQL Geoclustering (1und1 Internet AG)

Der Kunde hat seine Hosting-Infrastruktur auf Geoclustering *(Transatlantik-Spiegelung)* erweitert, um seine Ausfallsicherheit und permanente Verfügbarkeit weiter zu erhöhen.

Um bei einer Umschaltung einzelner Clusterknoten unnötigen transkontinentalen Quertraffic zu vermeiden, war es nötig, einzelne Kunden-Datenbanken zwischen den Clusterknoten zu verschieben *(Entmaschung)*, ohne jedoch den betreffenden Kunden neue Zugagngsdaten zuteilen zu müssen.

Um dieses Ziel zu erreichen, wurde ein transparenter Umleitungsmechanismus implementiert, mittels dem der Kunde seine Datenbanken auch nach Umzug unter denselben Zugangsdaten erreicht. Hierzu waren auch tiefergehende Eingriffe in der MySQL-Software wie auch diversen Inhouse-Systemen erforderlich.

Dem vorraus gingen eingehende Analysen, um sicherzustellen, daß die gewählte Lösung sowohl technisch sauber funktioniert, aber auch in der einzigartigen Größenordnung beim Massenhosting *(mehrere Millionen Kunden)* sicher und performant betrieben werden kann.

Parallel dazu waren noch Probleme mit DRBD zu lösen, um eine sicherere und performante Transatlantik-Spiegelung *(auf Petabyte-Skala)* zu ermöglichen.

Diese Arbeiten nahmen ca. 4 Monate in Anspruch.

Tools+Spachen: C/C++, PHP, Perl, MySQL, GNU/Linux (incl. Kernel-Entwicklung), Sybase, Git, GCC, Debian Linux, Subversion, iptables, OpenVZ, DRBD

2.2 (IOT) Beratung/Entwicklung Embedded-Linux/Diagnose - Bahntechnik, Hochsee-Schiffahrt, Kraftwerke

Der Kunde ist Hersteller von Sensorik und Meßtechnik. Das Projekt befaßte sich mit der OS-Plattform und SW-/System-Architektur eines Telemetrie-/Diagnose-Geräts *(vorrausschauende Wartung)* für rotierende Kraftmaschinen, insbesondere große Motoren und Turbinen. Es wird vorallen in den Bereichen Bahntechnik *(Lokomotiven)*, Seefahrt *(Schiffsmotoren-/Aggregate)*, Kraftwerke *(zB. Gas-Turbinen)* eingesetzt, um frühzeitig etwaige Maschinen- Schäden, lange vor einem Ausfall, erkennen zu können.

Wichtige Anforderungen sind hier uA. hohe Robustheit/Ausfallsicherheit und lange Lebenszeit unter rauhen Bedingungen *(zB. auf hoher See, in Kraftwerken, etc)* geringer Wartungsaufwand, Umgang mit schlechten Datenverbindungen. Hinzu kommen vielfältigste kundenspezifische Anpassugen.

Kernaufgaben im Projekt waren uA.:

- Einführung von auf Langzeitpflege und große Variantenvielfalt ausgelegte SCM-Techniken
- robuste vollautomatische Build-/Deployment-Prozesse *(DevOps)*
- Kernel-Portierung / Treiberentwicklung für div. Controller *(uA. Duagon IONIA)*
- Konsolidierung der HW-Ansteuerung auf Linux-Standard-Subsysteme *(zB. IIO)*
- Beratung zum Configurations- und Lizenzmanagement
- Portierung von nodejs für embedded systems
- Beratung zur SW-Architektur und Entwicklungs-Prozessen
- Beratung zur Chipset-Auswahl / HW-Architektur für Custom Boards
- Evaluation div. HW *(zB. NI cRIO)* auf SW-technische Nutzbarkeit für GNU/Linux-Systeme
- Aufbau einer Buildfarm / CI mittels Jenkins, Docker, Kubernetes

Tools+Spachen: C/C++, Linux-Kernel, PtxDist, Nodejs, Barebox, ADC, Git, RS485, CAN, RS232, Duagon IONIA, MVB, cmake, javascript, Docker, Kubernetes

2.3 Security-Beratung - Android - Anti-Tampering

- Beratung zu Android-Security im Zahlungsverkehr/Banking
- Anti-Tamperating / Integritätsprüfung innerhalb Android-Apps

Tools+Spachen: C/C++, GNU make, Linux, Android, Java

2.4 (DevOps) Ansible Modulentwicklung

- Entwicklung kundenspezifischer Ansible-Module
- Provisionierung von edge computing Systemen

Tools+Spachen: Git, Ansible, Python, Debian packaging, DevOps, Edge Computing, Embedded

2.5 Kernel-Treiber / BSP für PCEngines APUv2/APUv3

- Treiber-Entwicklung für AMD G-series (GX-412TC) GPIO
- Treiber-Entwicklung für PC-Engines APUv2 / APUv3 (LEDs + Buttons)
- Integration in die Mainline
- offizieller Kernel-Maintainer
- Build-Umgebung + Debian-Paketiereung
- Debian-basiertes Board Support Package
- Testautomatisierung

Tools+Spachen: C/C++, GNU make, Linux-Kernel, Debian, Devuan, Jenkins, Git, Perl, Python, Shellscript, GNU/Linux

2.6 (IOT) Beratung/Entwicklung Embedded-Linux Gebäudetechnik

Der Kunde ist internationaler Hersteller von Gebäudeautomatisierungen, insbesondere Aufzugstechnik. Für diesen Kunden wurden zwei Projekte bzw. Geräte betreut: einerseits Aufzugssteuerung, andererseits ein externes Telemetriegerät für Monitoring und ”predictive maintenance”.

Aufzugssteuerung

- Automatisierung und Optimierung von Build- und Deployment- Prozessen und Source-Control-Management
- Beratung bzgl. Software-Archiktetur und Entwicklungsstrategie
- Schulung der Kollegen bzgl. Entwicklung unter GNU/Linux
- Code-Reviews und Cleanup

Telemetriegerät (IOT)

In diesem Projekt wurde ein vorhandenes Telemetriegerät um weitere Funktionen, insbesondere ”Aufzugswärter” *(externes Monitoring des Steuergeräts)* und Notrufsystem, aber auch Anbindung weiterer Steuergeräte- Modelle, erweitert. *(uA. zur Ablösung auslaufender Zusatzgeräte)*

- BSP- und Kernel-Anpassungen
- Refactoring und Optimierung zur Nutzung von GNU/Linux-Bordmitteln *(statt Legacy-Code)*
- Refactoring der HW-Ansteuerung / Treiberschicht
- Refactoring der Techniker-Funktionen
- Refactoring der Cloud-Kommunikation
- Implementierung von Aufzugswärter- *(Monitoring/Selftest)* und Notruf-Funktion

- Anbindung von Modem/GSM-TA
- Anbindung weiterer Steuergeräte
- Harmonisierung auf internationale Gegebenheiten *(div. Gehäuse/LED-Panel, Bedienfunktionen, etc.)*
- Architektur- und Code-Review

Tools+Spachen: C/C++, Linux-Kernel, PtxDist, Azure, Git, Perforce, Redmine, LaTeX, RS485, CAN, RS232, cmake, javascript

2.7 Automotive: Security-Beratung

- Security-Beratung für vernetzte Automobil-Komponenten
- Software-Architektektur und Reviews

Tools+Spachen: C/C++, GNU make, Linux, Android, Java

2.8 CAN/RS485 datalogger (Maschinenbau / IOT)

Für einen Kunden aus dem Maschinenbau wurde ein flexibles Telemetrie/Datalogger- Gerät benötigt, das Betriebsdaten zwecks späterer Analyse / Monitoring aufzeichnen und später abrufbar macht bzw. automatisiert auf einem Server einliefert.

Die Geräte sind zuweilen im Feld *(weltweit verteilt)* schwer zugänglich, oft auch mobil, und verfügen zuweilen nur sporadisch über Datenverbindungen. Deshalb mußte sowohl auf geringen Energieverbrauch, aber auch Kompression und sehr robuste Datenübertragung geachtet werden.

Kernpunkte des Projekts waren:

- Minimales Linux-System *(PtxDist)* mit kundenspezifischem BSP *(incl. Kernel-/Treiber-Entwicklung)*
- Logging über verschiedene Schnittstellen *(zB. CAN, RS485, RS232, MODBUS, IP)* und kundenspezifischen Protokollen
- ggf. Sonderlösungen zur Datenanbindung *(zB. AX25 / packet radio)*
- Logging auf interne microSD und Upload via UUCP und RSYNC.
- Vollautomatisches Build, Deployment, Field-Updates
- Konfigurations-Management
- Factory-Tools

Tools+Spachen: C/C++, Linux-Kernel, Barebox, PtxDist, Git, CAN, RS232, RS485, UUCP

2.9 Datenbank-Konzeption und RADIUS für DSL-Provider

Der Auftraggeber benötigte eine Kundendatenbank für seinen DSL-Vertrieb. Sämtliche Aktionen wurden auch per Web zugänglich gemacht, sodaß der Kunde seine Bestellung, Tarifwechsel, Kontrolle und Pflege der Stammdaten bequem per Web erledigen kann.

Aus der Datenbank heraus wird auch der Radius-Server betrieben.

Tools+Spachen: C/C++, PHP, Perl, PostgreSQL , plsql, CVS, Radius, X.75, DSL, IP tunneling, Shellscript, LaTeX, LAMP, GNU/Linux, CISCO

2.10 Vollautomatischer HIL-test via CI-RT und Labgrid

- Integration von CI-RT, Labgrid, Jenkins, Docker
- Vollautomatische Provisionierung der CI-Umgebung via Docker
- Tool-Paketierung für ELBE

Tools+Spachen: C/C++, GNU make, Linux-Kernel, ELBE, Debian, Jenkins, CI-RT, Git, Perl, Python, Labgrid, Shellscript, GNU/Linux

2.11 Portierung von PostgreSQL auf Collax Linux Appliance

Der Kunde *(anonym)* ist Collax-Partner und betreut mittlerweile eine Vielzahl von Kunden mit Collax-Servern *(einer Linux-basierten Business-Server Appliance mit einfach bedienbarem Web-Interface)*.

Trotz der Vielzahl der Funktionen mangelt es dem Collax noch an einem relationalem Datenbank-System. PostgreSQL bietet sich hier aufgrund des hohem Funktionsumfangs und guter Performance an.

Das PostgreSQL -Paket für Collax habe ich mit meinem selbst entwickelten *metux Briegel Builder* und einem Crosscompiler erzeugt und Patches für das saubere Crosscompiling geliefert.

Tools+Spachen: C/C++, Perl, Javascript, Crosstool, Briegel , PostgreSQL , GNU make, GNU autotools, Subversion, GNU/Linux, Collax Business Server

2.12 Container-Lösnug für Embedded/IOT/Edge

- Container-Lösung für Embedded/IOT/Edge-Devices
- Crosscompiling-Umgebung für Google Go
- Portierung von conainerd *(Docker)*
- Integration in CI/CD/HIL

Tools+Spachen: C/C++, GNU make, Linux-Kernel, Debian, Jenkins, Docker, Go, Git, Python, Labgrid, Shellscript, GNU/Linux

2.13 Webcrawler / Scanner und Mailgate für Web-Foren

Bei seiner täglichen Arbeit benötigt der Kunde stets aktuelle Informationen aus vielerlei Web-Foren, Newsportalen, udgl. Als Kommunikationsmedium bevorzugt der Kunde E-Mail.

So lautet hier die Aufgabenstellung, Scanner-Anwendungen zu programmieren, die verschiedene Portale und Foren regelmäßig abfragen, Information extrahieren und die **neuen** Nachrichten per Email weiterleiten. Für die Foren gibt es zudem umgekehrt auch einen Email-Roboter, der empfange Mails wieder ins Forum schreibt - so erhält man den Komfort einer Mailingliste bzw. Newsgroup.

Tools+Spachen: C/C++, PHP, PostgreSQL , Apache HTTPd, CVS, GNU make, GCC, Shellscript, LAMP, GNU/Linux

2.14 Crosscompile-Umgebung für Rust (Embedded)2019

- Crosscompiling-Umgebung für Rust
- Anbindung an PtxDist-Buildumgebung
- Integration in CI/CD/HIL

Tools+Spachen: C/C++, GNU make, Rust, Debian, Jenkins, Docker, Go, Git, Python, Labgrid, Shellscript, GNU/Linux

2.15 Embedded-Linux Entwicklungsplattform und Paket-Portierung

Der Kunde *(aufgrund NDA nicht näher genannt)* entwickelt eine ISP-Plattform zum Aufbau von öffentlichen WLAN-Netzen mittels gängigen DSL-Anschlüssen und den dort verfügbaren Endgeräten.

Dazu wird für die gängigen DSL-WLAN-Router *(zB. FritzBox)* eine eigene Linux-basierte Firmware mit einer Virtualisierungslösung benötigt. Existierende Distributionen/Werkzeuge *(zB. OpenWRT)* wurden für ungeeignet befunden.

Ich habe für dieses Projekt mein Produkt *metux Briegel Builder* geliefert, das sämtliche Build-Schritte - vom Herunterladen der Quellen, Patching, Compilieren in einer virtuellen Umgebung *(SYSROOT)* bishin zur Paketierung und Image-Generierung - vollautomatisch ausführt.

Dazu gehören zahlreiche Anpassungen und Fixes an den verwendeten Paketen *(zB. um ein sauberes Crosscompiling zu ermöglichen)*, für die von mir Patches geliefert wurden.

Tools+Spachen: C/C++, GNU make, Linux-Kernel, MPLS, Crosstool, Briegel, Diffutils, CVS, Perl, Shellscript, GNU/Linux, metux Briegel

2.16 Debian Packaging: R / Shiny

- Automated build and packaging of R/SHINY for Debian-based Distros
- Fixed upstream's broken build process
- Integration into Debian's nodejs, R, service infrastructure

Tools+Spachen: Debian, R, dpkg, APT, docker-buildpackage, GNU make, Cmake

2.17 Kernel-Treiber / BSP und SW-Architektur für Outdoor display panels

- Treiber-Entwicklung / BSP für iMX.6-basierte Display-Boards
- Build-Umgebung + Debian-Paketiereung
- Debian-basiertes Board Support Package
- Testautomatisierung
- Software-Architektur
- Surf-Browser basierte display server software *(Content vom Datacenter bespeist)*

Tools+Spachen: C/C++, GNU make, Linux-Kernel, PtxDist, Git, Perl, Python, Qt, Webkit, Surf, Shellscript, GNU/Linux

2.18 (DevOps) Docker application container

Viele größere Anwendungen liegen in den üblichen GNU/Linux Distributionen nur in älteren Versionen vor. Die direkte Paketierung solcher Anwendungen, gerade für ältere *(LTS)* Distro-Versionen, gestaltet sich oft aufgrund zahlreicher Abhängigkeiten sehr aufwändig.

Eine Alternative bilden Application Container:

- dedizierte Container-Images pro Anwendung
- basierend auf minimalen Basis-Distros *(zB. Alpine oder Devuan)*
- Start-Scripte im Host, welche die jeweilgen Anwendungen – transparent für den Nutzer – starten und in die gewohnte Desktop-Umgebung integrieren
- Jenkins-Pipeline für vollautomatische builds

Zu den containerisierten Anwendungen zählen zB. LibreOffice, GIMP, LaTeX, Skype, Chromium, Thunderbird, Firebird, golang, etc.

Tools+Spachen: dpkg, apt, docker-buildpackage, Docker, Jenkins, GNU make, Shell, Debian, Devuan, LibreOffice, LaTeX, Skype, Thunderbird, Firebird, Golang, Chromium

2.19 (DevOps) Docker-Buildpackage: Build-Automatisierung für Debian-/Ubuntu-Pakete

- Tool zum einfachen und vollautomatischen Build von deb-Paketen und Paket-Repositories
- strikt isolierte Build-Umgebung via Docker
- deutlich leichter aufzusetzen als bisherige Tools *(zB. pbuilder oder buildd)*
- dient uA. der Pflege kundenspezifischer oder auch eigener LTS-Repositories
- einfache Integration in CI *(zB. Jenkins)*

Tools+Spachen: Debian, APT, dpkg, Docker, docker-buildpackage, CI, Jenkins, DevOps

2.20 (DevOps) Paketierung von Docker + Kubernetes für Debian/Devuan

Docker und Kubernetes sind aktuell nur unzureichend für Debian bzw garnicht paketiert. Die Software-Architektur und Build-Prozesse sind inkompatibel mit den Mechanismen der GNU/Linux Distributionen.

Der Upstream geht davon aus, daß lediglich *(wie bei proprietärer Software auch)* – komplett an der Distro vorbei – als Binary heruntergeladen und manuell installiert wird. Schlimmer noch: Kubernetes installiert/aktualisiert seine Komponenten selbstständig – an Toolchain und Qualitätskontrolle der Distro vorbei.

Hierdurch leiden Wartbarkeit und Qualität *(insbesondere auch im Bezug auf automatische Sicherheitsupdates, aber auch Integration in das Restsystem)* erheblich – es entstehen nicht unerhebliche Extra-Kosten und -Risiken.

Deshalb ist es erforderlich, die beiden Tools mit den zugehörigen Distro- Toolchains für deb/apt zu paketieren.

- Debian-Paketierung aktueller golang-Toolchain
- Debian-Paketierung zahlreicher go-Module *(Abhängigkeiten)*
- Refactoring des Build-Prozess von Docker und Kubernetes *(zB. Build mit Distro-Toolchain)*
- Debian-Paketierung von Docker und Kubernetes
- SCM-Infrastruktur / -Workflows für die Langzeitpflege
- Generieren von APT-Repositories für Golang, Docker, Kubernetes
- Jenkins-basierte CI

Tools+Spachen: dpkg, apt, docker-buildpackage, Go, Docker, Kubernetes, debhelper, Jenkins, GNU make

2.21 IT-Outsourcing für Ärzteverbände

Die beiden Ärzteverbände ”Deutsches Forum für Psychotherapie” *(DPI)* und die ”Vereinigung psychotherapeutisch tätiger Kassenärzte” *(VPK)* haben metux IT service ihre komplette EDV überantwortet.

Neben alltäglicher Wartung von EDV-Anlagen und redaktionellen Diensten zählt hierzu eine ASP-Lösung *(Anwendungsdienst)*, mit der die Verbände ihre Mitglieder verwalten, Beiträge berechnen und sich in zahlreichen geschlossenen und offenen Mailinglisten austauschen können.

Zudem wurde eine Lösung erarbeitet, mit der die Verbände ihre Rundschreiben via Web erstellen, in Druckereiqualität automatisch setzen und anschließend sowohl als Newsletter ebenso wie als Briefpost verteilen lassen können.

Tools+Spachen: PHP, Perl, PostgreSQL , Javascript, LaTeX, Majordomo, Java, CVS, Shellscript, LAMP, GNU/Linux

2.22 (IOT) Kernel / BSP für Duagon IONIA (Automotive, Bahntechnik)

Für einen Kunden aus der Bahntechnik wurde ein neues Linux-BSP für Duagon IONIA Maschinen entwickelt. Die mitgelieferte/vorinstallierte Software war für den Anwendungsfall *(als generischen Industrie-Rechner anstatt MVB-Controller)* nicht geeignet, uA. wegen:

- sehr veraltet *(uA. viele fehlende Kernel-Features)*, sehr geringe SW-Auswahl
- fehlende Reproduzierbarkeit aus dem Sourcecode heraus *(Duagon selbst konnte das BSP nicht mehr komplett recompilieren, geschweige denn Komponenten wie zB. Kernel upgraden)*
- proprietäre Treiber-APIs, die umfangreiche HW-spezifische Anpassungen in den Anwendungen nötig machen - statt Standard-APIs *(zB. IIO)*
- zudem nur auf single-process / root only ausgelegt – sehr problematisch bzgl. Stabilität und Sicherheit
- keine robusten / vollautomatischen Deployment-Mechanismen *(Paketmanagement, etc)* fehlt.

Das Projekt beinhaltete uA:

- Portierung des aktuellem Mainline-Kernels *(4.x)* für das IONIA CPU-Modul *(Treiber, Devicetree, Config, ...)*
- Kernel-Treiber: backplane/serial ports, RPC, DIO, LOG, IO, I701, I202, I012, IIO backend, etc.
- Basis-System / BSP *(ptxdist-basiert)*
- Factory-Tools
- Beratung/Schulung zu IIO

Tools+Spachen: C/C++, Linux-Kernel, PtxDist, ADC, Git, CAN, RS232, Duagon IONIA, MVB, cmake

2.23 (IOT) Lizenzmanagement für Embedded Devices (Automotive, Bahntechnik)

Der Kunde ist Hersteller von Telemetrie-/Diagnose-Lösungen, welche in großen Transportmaschinen und Industrieanlagen *(Lokomotiven, Schiffe, Kraftwerke, etc)* weltweit eingesetzt werden. Hierbei lassen sich unterschiedliche Funktionalitäten separat buchen. Seinerseits setzt der Kunde Zulieferkomponenten ein, welche analog je nach benötigtem Funktionsumfang lizensiert werden.

Kernpunkte hierbei sind:

- Modellierung der Features – seitens des Produkts, als auch der Zulieferkomponenten – nebst ihrer Abhängigkeiten
- Konzeption einer CMDB zur Verwaltung von Geräten, gebuchten Features, Lizenzen, etc.
- automatisches Generieren der Geräte-Configurationen
- Cryptographische Absicherung und Secure Boot
- Reporting-Werkzeuge
- ERP-Anbindung

Tools+Spachen: PtxDist, OpenSSL, GnuPG, Packaging, Jenkins, imx6, trusted boot, PostgreSQL, ERP

2.24 Embedded: Spriometrie- und EKG-Meßcomputer für klinische Studien

Der Kunde ist führender Anbieter für die technische Abwicklung klinischer Studien zur Medikamentenzulassung im Bereich Spirometrie *(Lungenfunktionsmessung)* und EKG.

Hierzu stellt er den auftraggebenen Pharma-Herstellern uA. spezialisierte Meßcomputer mit jeweils studienspezifisch angepaßter Firmware zur Verfügung. Diese Geräte werden in den an den Studien weltweit teilnehmenden Kliniken aufgestellt und vom dortigen medizinischen Personal zur Datenerfassung, Messung verwendet. Anschließend werden diese gesammelten Daten ins Rechenzentrum übertragen, weiterverarbeitet *(Data-Cleaning)* und dem Pharma-Hersteller zur Studienauswertung übergeben.

Mein Verantwortungsbereich lag in der Weiterentwicklung und Anpassung des Produkte "FlowScreen" *(Spirometrie)* und "CoreScreen" *(EKG)*, sowie der Zusammenführung der verschiedener Produktlinien zu einem generischen, leicht anpaßbaren Gerät. Es handelt sich hierbei um einen ARM-MX.21-basierten Computer mit eingebauter Industrie-Tastatur, Touchscreen, Drucker *(HP OJ-470)*, mit einer Embedded-Linux-basierten Firmware.

Die erste Projektphase bestand in der Portierung der Software auf eine neue Hardware-Plattform. Insbesondere mußten hier Toolchain, Buildprozesse, Bootup, Treiber, udgl. angepaßt/portiert werden. In dem Zuge habe ich auch das Firmware- Deployment derart vereinfacht, daß Upgrades nun auch vom Benutzer im Feld selbst durchgeführt werden können.

In der zweiten Projektphase wurden dann die ersten Studien-Entwicklungen *(kundenspezifische Customizations)* auf der neuen Plattform durchgeführt. Zum vereinfachten Managemant derartiger paralleler bzw. aufeinander basierender Entwicklungslinien habe ich zudem neue SCM-Prozesse und Tools *(Git-basiert, downstream-branches, etc)*, sowie automatisiertes Build-/Delivery *(contigious integration, etc)* eingeführt. Desweiteren habe ich auch die Entwicklungs-Infrastruktur modernisiert, um diese robuster und um ein vielfaches schneller zu gestalten.

Mit der dritten Projektphase wurden, parallel zu verschiedenen Studien-Entwicklungen wurde die Software-Plattform weitgehend modernisiert, insbesondere die vormals jeweils direkt im Quellcode stattgefundenen kundenspezifische Anpassungen durch generische Configuration/Parametrisierung abgelöst. Damit wurden die Aufwände und Zeitrahmen für die studienspezifischen Anpassungen im Schnitt um ca. 2/3 reduziert.

Insgesamt nahm das Projekt ein volles Jahr in Anspruch.

Tools+Spachen: C/C++, Perl, GNU/Linux (incl. Kernel-Entwicklung), Git, GCC, Crosscompiler, ARM/MX.21

2.25 Echtzeit-Börsenhandelsplattform im Devisenmarkt / FOREX

Der Kunde betreibt eine Plattform für den automatisierten Echtzeithandel am Devisenmarkt *(FOREX)* mit variablen Hebeln.

Hierzu bündelt die fXignal-Plattform Handelssignale aus verschiedensten Quellen - eigene Tradestations, externe Analysten oder manuelle Eingabe - und gruppiert diese nach verschiedenen Handelsstrategien und Währungspaaren in separate Kanäle. Der Nutzer kann diese Kanäle mit diversen Parametern *(Limits, Lot, etc)* auf seine Broker-Konten leiten und so die fXignal-Plattform vollautomatisch handeln lassen.

Die Plattform war zunächst historisch aus dem eigenen Handelsgeschäft des Kunden gewachsen und wurde später für weitere Nutzer geöffent. Mit dem weiteren Wachstum hatte sich diese gewachsene Struktur aus zahlreichen als Windows- und GNU/Linux-Systemem mit verschiedenen inkompatiblen Anwendungen als wenig zukunftsträchtig herausgestellt und mußte im laufenden Betrieb - unterbrechungsfrei - neu strukturiert werden.

Nach Zulieferung einiger Software-Komponenten und verschiedener Operating-Tasks übernahm ich mit Ende 2004 hier die technische Leitung von der Weiterentwicklung und Neustrukturierung der Plattform bishin zum Operating des Rechenzentrums. Wichtige Meilensteine waren hier:

- Restrukturierung und Optimierung der Datenbank-Infrastrukturen im laufenden Betrieb, um den Echtzeit- und Stabilitäts-Anforderungen gerecht zu werden
- Ablösung der MS-SQL-Server und weiterer damit verbundener Windows-Systeme durch PostgreSQL auf einer eigens entwickelten, optimierten GNU/Linux-Appliance

- Ablösung des Windows-/IIS-Webservers durch Apache HTTPd auf einer eigens entwickelten, optimierten GNU/Linux-Appliance
- Ablösung der Windows-/Exchange-Mailserver durch Sendmail und diverser Web-Frontends auf LAMP-Basis, auf einer eigens entwickelten, optmierten GNU/Linux-Appliance
- Ablösung verschiedenster Delphi-basierter Dienste, teils durch Stored-Procedures, teils durch Java- und C/C++-basierte Daemons, um den Echtzeit-Anforderungen gerecht zu werden
- Implementation eines echtzeitfähigen Reportings
- Ablösung des C#-basierten *(Windows-)*Clients durch eine neu entwickelte Nutzerplattform auf PHP-Basis
- Implementation eines mobilen Web-Frontends
- Ablösung der Windows-basierten Orderstations *(Broker-Ansteuerung)* durch durch einen GNU/Linux-basierten Order-Robot
- Anbindung weiterer Signalquellen und Broker *(Refco, FXCM, etc)*
- Virtualisierung der verbleibenden Windows-Systeme *(Tradestations)* für einfacheres Management und Kostenreduktion
- Schrittweise Implementation eines Hot-Standby für kritische Services
- Restrukturierung der RZ-Verkabelung, Redundante Anbindung und Core-Router *(CISCO+GNU/Linux+Lucent-Mischumgebung, AS/BGP via Zebra)*
- Restrukturierung der VPN-Infrastruktur

Tools+Spachen: C/C++, PostgreSQL , MS-SQL, IIS, Apache, Zebra, PHP, Java, CISCO, Tradestation, LAMP, Sendmail, GNU/Linux, Briegel

2.26 (IOT) Maßgeschneiderte GNU/Linux-Firmware für WLAN-VoIP-Router

Die TU-Ilmenau entwickelte in einem Forschungsprojekt eine Infrastruktur für mobile Telefonie- und Videodienste auf Basis von gängier WLAN-Technik. Es werden dabei sowohl die Zellrouter, das MPLS-basierte Traffic-Management-System und Referenzfirmware für mobile Endgeräte entwickelt.

Meine Aufgabe bestand vorallem darin, eine möglichst schmale und optimierte GNU/Linux-basierte Firmware auf Grundlage welteit entwickelter Komponenten zu implementieren und zukünftig auch ständig auf dem neuesten Stand zu halten. Gängige Distributionen sind hier aufgrund der speziellen Systemanforderungen nicht geeignet.

Dank meines selbst entwickelten *metux Briegel Builder* war diese Aufgabe - im Vergleich zu gewohnter Handarbeit - recht unkompliziert und rasch zu meistern.

Viele der verwendeten Pakete waren noch nicht für Embedded-Systeme und Crosscompiling geeignet. Hierzu habe ich entsprechende Patches geliefert.

Wichtig war jedoch auch die Anpassung und Weiterentwicklung des Linux-Kernels. Beispielsweise waren zu diesem Zeitpunkt die MPLS+RSVP-Implementation, wie auch verschiedene WLAN-Treiber noch nicht produktiv einsetzbar. Hier habe ich dem Kunden einen produktiv benutzbaren Kernel geliefert.

Tools+Spachen: C/C++, Perl, Java, MPLS, WLAN, Intel Geode, Crosstool, Briegel, GNU make, GNU autotools, diffutils, CVS, Subversion, Shellscript, GNU/Linux, metux Briegel

2.27 Automatisierte Prüfung von Meßgeräten

Der Kunde ist Sachverständiger für Kalibrierung von Prüfung von Druckmeßtechnik. Es wurde eine Softwarelösung für die Erfassung der Meßdaten, Erstellung von Prüfberichten sowie der Verwaltung von Kundendaten, Kundengeräten und Prüfmitteln entwickelt.

In enger Zusammarbeit wird das Produkt nun auch für andere Meßgerätearten (z.b. Längenmeßtechnik) ausgebaut und in Kürze als Branchenlösung auf dem Markt angeboten.

Tools+Spachen: PHP, PostgreSQL , Apache, LaTeX, Shellscript, LAMP, GNU/Linux

2.28 Java/JNI-Anbindugn der Linux Kernel Crypto API

- Einbindung der Linux Kernel Crypto API in Java Virtual Machine zwecks besserer Performance und Sicherheit
- Build-Umgebung / BSP *(yocto)*

Tools+Spachen: C/C++, Java, JNI, GNU make, Linux-Kernel, Yocto, Git, Shellscript, GNU/Linux

2.29 (IOT) HW-Anforderungen und Embedded-Linux BSP (Bahntechnik)

- Entwicklung eines Board Support Package für ein Bahntechnik-Gerät *(predictive maintenance)*
- HW-Systemanforderungen und Beratung zur Chip-Auswahl
- Spezifikation der Schnittstelle zwischen CPU und ADC-Baugruppe/FPGA
- Anpassung von Kernel und Bootloader, Treiber-Entwicklung
- automatisierte Build-Umgebung via PtxDist und Jenkins
- Factory-Tools

Tools+Spachen: C/C++, Linux-Kernel, Barebox, PtxDist, ADC, Git, CAN, RS232, Bitbucket, Jenkins, DevOps

2.30 Entwicklung Server für verschlüsselte Mailverteiler/Mailinglisten

- Maillist-Server mit eingebauter GPG-Verschlüsselung / Umschlüsselung
- eingehender Traffic mit Public Key der Liste verschlüsselt
- Listserver verschlüsselt für einzelne *(Ziel-)*Teilnehmer neu
- Rechteverwaltung via Crypto-Tokens *(signierte Kommandos)*
- Verschlüsselte und Signierte Archiviertung im Venti-Store
- Strikte Komponenten-Separation via Namespaces/Container

Tools+Spachen: C/C++, Gnupg, Python, GCC, qmail, GNU/Linux, Plan9Port, Venti, Plan9 secstore, SMTP

2.31 (IOT) HW-Anforderungen und Embedded-Linux BSP (Bahntechnik)

Für einen Kunden aus der Bahntechnik wurde ein neues Mainboard nebst Linux-BSP entwickelt. Das Gerät dient der Messung von Rotationsmaschinen *(zB. große Dieser-Motoren)* zwecks Diagnose und vorrausschauender Wartung.

Das Projekt beinhaltete uA:

- HW-Systemanforderungen
- Spezifikation der Schnittstelle zwischen CPU und ADC-Baugruppe/FPGA
- Anpassung von Kernel und Bootloader
- BSP und Build-Umgebung
- Treiber-Entwicklung
- Factory-Tools

Tools+Spachen: C/C++, Linux-Kernel, Barebox, PtxDist, ADC, Git, CAN, RS232

2.32 Beratung/Entwicklung Embedded-Linux Medizintechnik

Der Kunde ist Hersteller für klinische Sterilisationsgeräte. Im Zuge des Projektes wurde die Plattform für die neuen Gerätegenerationen entwickelt.

Im Gerät befindet sich ein imx53-Rechner mit Touch-Panel, Netzwerkport *(zum Anschluß ans Praxisnetz)*, diverse USB-Ports *(Wartungsschnittstelle, Drucker-Anschluß, externer Speicher, etc)* sowie CAN-Ports zur Kommunikation mit weiteren Steuerprozessoren. Über dieses findet neben der Bedienung auch Datenverfassung und Ausdruck, Monitoring, SW-Updates, udgl. statt.

Hauptaugenmerk des Projekteinsatz bestand in der Bereitstellung einer Betriebssystemumgebung und Entwicklungsinfrastruktur für og. Bedienteil, bishin zu Deployment und Feld-Updates nebst Beratung bzgl. Wartbarkeit und Sicherheit.

Kernpunkte waren uA.:

- ptxdist-basiertes BSP *(Board Support Package)* nebst Treiberentwicklung/-Anpassung *(uA. auch Portierung von Qt5 auf KMS/DRI)*
- Build- und Deployment-Prozesse mit Hinblick auf Fertigung und Wartung im Feld
- allgemeine Entwicklungsrichlinien nebst Schulung der Kollegen
- Automatisierung von Dokumenten und Code-Generierung
- Sicherheitsanalysen *(insb. Manipulationssicherheit und Knowhow-Schutz)*
- Beratung bzgl. SW-Architektur
- cryptographisch gesicherte Wartungsschnittstelle
- Evaluation neuer Prozessor-/SOC-Generationen *(uA. IMX6)*
- Review und Abnahme der Zulieferungen externer Dienstleister

Tools+Spachen: LaTeX, C/C++, Linux-Kernel, ptxdist, Git, shellscript, python, cmake, GNU make, OpenGL/Mesa, Qt5, imx53, imx6

2.33 GNU/Linux Software Packaging und automatische Builds

Der Auftraggeber ist Anbieter für Business Cloud Lösungen und hat hier verschiedene Produkte, zB. Zimbra , Plone, Redmine, OpenERP im Portfolio. Er entwickelt diese auch weiter und paßt sie kundenspezifisch an.

Für ein einfaches und reproduzierbares Deployment habe ich hier eine stringente Paketierung eingeführt: jegliche Software *(auch proprietäre Produkte, wie zB. CrashPlan)* werden ausschließlich mittels Paketmanagement der jeweiligen Plattform ausgerollt - vom Entwicklungs-Testsystem über Validierung/Preproduction bishin zum tatsächlichen Produktivsystem.

Die verschiedenen Qualitätsstufen werden hier durch separate Repositories abgebildet.

Desweiteren wurde eine automatische Build-Infrastruktur *(basierend auf Jenkins und Kubernetes)* implementiert, welche die aktuelle Codebasis vollautomatisch compiliert, paketiert, auf Testsysteme ausrollt *(via Paketmanagement)* und verschiedenste Tests abfährt.

Mittels dieser Infrastruktur wurden die Rollout-Zeiten *(in allen Stufen, von der Entwicklung bishin zum Produktivbetrieb)* deutlich reduziert und ein hohes Maß an Reproduzierbarkeit hergestelt.

Tools+Spachen: git, dpkg, apt, pbuilder, docker-buildpackage, Jenkins, Kubernetes, rpm, yum, mock

2.34 Virtualisierte und Redundante Serverumgebung

Der Kunde betreibt mehrere Server mit verschiedensten Anwendungen *(Mailserver, Webserver, Datenbanken, eigenes Web-Portal)*, die für den Geschäftsbetrieb kritisch sind. Aufgrund eines bereits erfolgten Angriffs ist dem Kunden die Sicherheit besonders in den Mittelpunkt gerückt.

Um die Schadmöglichkeiten im Falle eines Angriffs auf ein Minimum zu reduzieren, wird das ganze System mittels OpenVZ virtualisiert: jeder Dienst erhält eine eigene virtuelle Umgebung.

Die Software innerhalb der einzelnen VZ's stammt nicht aus einer gängigen Distribution, sondern wird mittels der Eigenentwicklung *metux Briegel Builder (siehe Projekte)* für jede einzelne VZ speziell erzeugt. Dabei enthält jede VZ nur jeweils nur einen Service und ausschließlich den dafür benötigen Programmcode. Updates, regelmäßige Integritätsprüfung und automatische Wiederherstellung der einzelnen VZ's finden vom Host-System aus statt. Das Host-System selbst ist im Normalbetrieb nicht via Internet erreichbar.

Eine nicht zu unterschätzende Herausforderung besteht in diesem Projekt darin, sämtliche benötigten Pakete für Crosscompiling/Sysroot vorzubereiten - viele Pakete sind dafür noch nicht von Haus aus geeignet.

Tools+Spachen: C/C++, Java, Perl, Shellscript, Briegel , Sendmail, lighttpd, PHP, Apache, qMail, IMAP, POP3, LDAP, PostgreSQL , mySQL, Lighttpd, Nagios, Git, GNU/Linux

2.35 Neuaufbau einer Webradio-Community

Der Kunde ist ein Independent-Webradio. Aufgrund technischer Probleme mit dem auf einer sehr alten Joomla-Version basierenden Portal wurde ein Neuaufbau in Auftrag gegeben.

Das neue Portal basiert auf dem aktuellen Joomla-1.5 mit Community Builder und *metux MediaCloud* sowie zusätzlicher Erweiterungen, zB. Shoutcast-Integration, Chat, Forum, u.v.m.

Tools+Spachen: PHP, LAMP, metux MediaCloud, Joomla, CommunityBuilder, Shoutcast, GNU/Linux, Lighttpd

2.36 OPAL-Implementierung auf Linux-Embedded-Device (automotive)

- Implementierung von OPAL (SSD-Verschlüsselung) in Embedded-Devices
- Referenz-Testing für verschiedene SSD-Hersteller/-Modelle
- Kernel-Anpassungen / Treiber-Portierung
- Bugfixing in OPAL toolchain
- Testautomatisierung

Tools+Spachen: C/C++, GNU make, OPAL, sed-util, Linux-Kernel, Debian, Devuan, Jenkins, Git, Perl, Python, Shellscript, GNU/Linux

2.37 Optimierte vServer-Images für mass hosting

Der Kunde betreibt Massen-Hosting von OpenVZ-basierten virtuellen Linux-Servern. Dazu bietet er eine Reihe vorgefertigter und optimierter Images für die verschiedensten Anwendungsgebiete, zB.

- Nameserver mit BIND oder DJBDNS
- Mailserver mit Sendmail oder qMail, UUCP, IMAP und Web-Frontend
- Web-Appliances mit lighttpd *(zB. Joomla CMS, vTiger CRM, ...)*
- Freenet nodes

Dem Kunden war bei dem Projekt wichtig, daß auch dauerhaft Updates und ggf. neue / veränderte Appliances, auch für andere Hardware-Architekturen rasch und kostengünstig geliefert werden können.

Die Images wurden mit meinem selbst entwickelten *metux Briegel Builder* und verschiedenen Crosscompilern erzeugt - mit diesem Werkzeug ist der Pflegeaufwand für derartige Projekte äußerst gering.

Tools+Spachen: C/C++, Shellscript, m4, Bind, lighttpd, GNU make, GNU autotools, UUCP, Crosstool, Briegel , Sendmail, qMail, Joomla, vTiger, CVS, Subversion, GNU/Linux

2.38 Joomla Veranstaltungskalender- und Buchung für Wohlfahrtsverband (Parisat)

Der Kunde ist eine gemeinnützige Gesellschaft, welche operative Tätigkeiten für den Thüringer Wohlfahrtsverband erledigt. Einer der Hauptaufgaben sind Schulungsveranstaltungen für die Mitglieder angeschlossenen Wohlfahrts-Organisationen.

Traditionell wurden die Terminplanungen mittels Lotus-Notes erledigt und händisch in die Joomla-basierte Website eingepflegt. Buchungen fanden manuell via eMail, Telefon oder Post ab.

Um diesen Vorgang zu automatisieren wurde ein Veranstaltungskalender mit Buchungsfunktion in der Website implementiert, welcher einerseits in der Veranstaltungen, andererseits auch bei der Buchung fachspezifische Informationen verwaltet. Zur einfachen Integration verfügt die Lösung über Import-/Export-Schnittstellen, welche uA. auch als Datenquelle für verschiedene Abteilungen, wie zB. die Print-Redaktion fungiert.

Derzeit wird - nach vielen Jahren erfolgreichem Betrieb - ein Relaunch auf aktuelle Joomla-Version und Integration mit der beim Kunden verwendeten Groupware *(Exchange)* realisiert.

Tools+Spachen: PHP, Joomla, Git, iCal, SMTP, IMAP

2.39 Refacturing vom QMail Mailserver-Paket

- Paketierung des Mailservers Qmail *(vollautomatisches Deployment)*
- Longterm-Maintenance / Bugfixing
- Security-Audit und Fuzzing
- Key-Management
- Migration von Sendmail auf Qmail

Tools+Spachen: C/C++, GNU make, GCC, qmail, Subversion, LaTeX, GNU/Linux

2.40 HIL-Test/Werksprüfung: Gebäudetechnik / Embedded-Linux

Der Kunde ist Hersteller von Premium-Haustechnik.

In dem Projekt wurde mittels eines HIL-Testands sowohl die Werksprüfung, als auch Entwicklungstests automatisiert. Dies wurde durch eine Kombination von Selbsttests auf dem Gerät und externe Signalgeber/-Abnehmer realisiert.

Dabei enthalten uA.:

- Netzwerk-Schnittstellen *(Last- und Langzeittests, Stabilität, ...)*
- Display- und Touch: Kalibierung und Qualitätskontrolle
- Thermische Prüfung
- Prüfung von IOs.
- Test von Audio/Video-Streaming, Codec, GPU

Tools+Spachen: C/C++, Linux-Kernel, PtxDist, Git, RS485, CAN, RS232, cmake, Qt, GStreamer, Labview

2.41 (DevOps) Vollautomatisches Deployment von Eclipse / Sigasi-Studio

Der Kunde setzt verschiedene Eclipse-Versionen und Eclipse-basierte Tools wie zB. Sigasi Studio ein. Diese galt es **vollautomatisch** auf verschiedene Zielsysteme auszurollen. Jegliche manuelle Schritte gilt es hier zu beseitigen, um unnötige Störung der Entwickler zu elimieren und eine jederzeit vollständig reproduzierbare Arbeitsumgebung zu gewährleisten.

Ein Hauptproblem bei Eclipse liegt hier darin, daß die Installation und Updates auf manuelle Durchführung und Ablage innerhalb des Home-Verzeichnis der einzelnen Nutzers ausgelegt ist.

Dabei werden uA. auch User-Configuration mit Programmcode vermengt - eine saubere Trennung, wie zB. in der Unix-Welt seit Jahrzehnten bewährt, fehlt hier. Dies führt zu fehlender Reproduzierbarkeit und erheblichen Störungen, insbesondere beim Parallelbetrieb verschiedener Versionen.

Bei Sigasi-Studio kam erschwerend hinzu, daß dieses lediglich mittels interaktivem Installer zur Online-Installation angeboten wurde. Diese ist aber im Netzwerk des Kunden aufgrund von Sicherheitsvorgaben nicht möglich.

Zur Lösung des Problem habe ich folgende Maßnahmen ergriffen:

- Reverse-Engineering des Deployment-Prozess und Struktur der P2-Repositories *(die offizielle Dokumentation ist äußerst mangelhaft)*
- Scripte zur automatischen Spiegelung von P2-Repositories ohne GUI nebst Ablage in Git
- Scripte zum automatischen Ausfiltern von Komponenten, die bereits im Standard-Eclipse enthalten sind *(Sigasi bietet lediglich ein Bundle aus Eclipse und Sigasi-Erweiterungen an)*
- Analyse der Abhängigkeiten und tatsächlich benötigter Komponenten / Versionen
- Scripte/Makefiles zur Generierung eines finalen Installations-Image und Debian-Metadaten.
- Vollautomatisch Build-Pipeline via Jenkins und docker-buildpackage, inclusive Generierung direkt installierbarer APT-Repository.
- Einbindung der APT-Repository in Ansible-Scripte für die verschiedenen Zielsysteme.

Tools+Spachen: Debian, APT, Packaging, Ansible, Eclipse, Jenkins, Sigasi-Studio, Docker-Buildpackage, Docker

2.42 (IOT/Automotive) Beratung zu Entwicklungsmethoden im Embedded-Linux-Bereich

- Kunde ist Hersteller von Meß- und Erfassungstechnik für Tanklastfahrzeuge *(zB. Treibstoff, Milch, etc)*
- Konzept zur Modernisierung der Basis-Software und Entwicklungsprozesse, Build-/Deployment, DevOps, Langzeitpflege und Portierung auf neue Hardware
- Konzeption für hochverfügbare Datenbank-Infrastruktur

Tools+Spachen: Git, SVN, C/C++, Linux-Kernel, GNU make, USB, DevOps

2.43 Unix-Sicherheitstools

Entwickelt wurden einige Sicherheitstools, u.A. dem Produkt ”su-wrapper” welches fein granulierte Zugriffssteuerung von Befehlsausführungen einzelner Benutzer unter anderen Benutzer *(z.b. root)* gestattet.

Dieses Tool wird beispielsweise eingesetzt, um unprivilegierten Benutzern gezielt Aktionen zu gestatten die betriebssystembedingt nur dem Adminstrator *(root)* erlaubt sind *(z.b. öffnen und schließen von Netzwerkverbindungen, etc)*.

Tools+Spachen: C/C++, GNU make, GCC, CVS, LaTeX, Shellscript, Nagios, GNU/Linux

2.44 Netzwerk-Optimierung und Dokumentation

Die Projektaufgabe bestand darin, existierende Netzwerkinfrastruktur und Adreßzuteilungen zu erfassen, zu optimieren und anschließend für die Servicetechniker zu dokumentieren. Im Anschluß wurden auch Wartungspläne erstellt.

Tools+Spachen: LaTeX, PostgreSQL , PHP, X.75, X.21, Radius, SNMP, Nagios, LAMP, Apache-HTTPd, Nagios, GNU/Linux, CISCO

2.45 Media-Portal Plattform: MediaCloud

Der Kunde betreibt seit einigen Jahren eine Community-Plattform für Künstler im Filmbereich *(Schauspieler, Filmemacher, udgl.)* auf Joomla-Basis, die um Video-/Musik-Upload erweitert werden sollte. Gleichzeitig sollten mehrere neue Video-Communities für verschiedenste Zielgruppen aufgebaut werden, die untereinander vernetzt sind und auf einen gemeinsamen Datenpool zugreifen.

In diesem Zuge wurde das Produkt *metux MediaCloud* entwickelt. Es handelt sich hierbei um eine eigenständige Server-Infrastruktur für größere Video-/Medien-Portale, die in verschiedenste Portalsysteme, wie zB. Joomla, eingebunden werden kann. Die *metux MediaCloud* -Infrastruktur übernimmt dabei Upload, Convertierung, Verwaltung und Wiedergabe der gesamten Mediendateien. An eine MediaCloud-Instanz können beliebig viele Portale mit individuellen Berechtigungen angebunden werden, sodaß beispielsweise Medien eines Portals in anderen sichtbar gemacht werden können.

Im Rahmen des Projektes wurde zunächst die Server-Infrastruktur errichtet/beutreut und sukzessiv 32 derartige Portale für verschiedene Zielgruppen mit diversen Zusatzfunktionen aufgebaut, beispielsweise:

- Gemeinsamer Userpool der beiden Hauptportale
- Roaming zwischen allen Portalen
- externes Roaming via OpenID
- Zahlungssystem / Verkauf von kostenpflichtigen Medien
- Shop-Integration
- Orderprozeß für Kreditkarten und Mobilfunkverträgen
- Portalspezifische Community-Anpassungen
- Datingportal mit Partner-Matching und Video-Chat

- Immobilien-Börse mit Videos
- Web-Phone-Integration *(Purtel)*
- Integration von Livestreams zahlreicher internationaler TV-Stationen
- Anbindung an Facebook und MySpace
- u.v.m.

Tools+Spachen: C/C++, Plan9, Venti, PHP, Perl, PostgreSQL , Javascript, AJAX, Shellscript, Lighttpd, LAMP, Git, ffmpeg, mplayer, Flash, Red5, Nagios, Gentoo Linux, metux MediaCloud

2.46 (DevOps) Vollautomatisches Deployment virtueller Entwicklungsumgebungen

Für einen Sensorik-Hersteller wurde das Deployment virtualisierter Entwicklungsumgebungen unter GNU/Linux *(Debian / Ubuntu)* vollständig automatisiert. Diese werden sowohl als VM-Images, Docker-/LXC-Container als auch direkt in vorhandene Systeme installiert.

Hierbei wurde auch die Deployment zahlreicher kommerzieller Software, welche sonst manuelle Installation via GUI erfordert, vollständig automatisiert.

- vollautomatische VM Provisionierung / Image-Erzeugung via Docker und Vagrant, Integration in ESXI-Umgebung, Integration in Win7 Desktop-Umgebung des Kunden
- vollautomatisches SW-Deployment und System-Configuration via Ansible
- Deployment/Provisionierung von Multi-User Workstations / VNC-Servern
- Build-Prozesse und automatisches Testing via Jenkins CI
- Archivierung in Artifactory und Git-Repositories
- Paketierung diverser Zusatz-SW für APT *(VTK8, Sigasi Studio, Vivado, ...)*
- Lizenz-Prüfung *(zB. Einhaltung div. FOSS-Lizenzen)*
- Build-Farm / CI auf einem Kubernetes-Cluster

Vom Kunden eingesetzte kommerzielle Software läßt sich von Haus aus nicht vollautomatisch und reproduzierbar via Paketmanagement installieren, sondern erfordern aufwändige manuelle Installation via GUI. Zuweilen verlangt der GUI-Installer auch Internet-Zugriff oder erfordert imensen temporäten Speicherplatz *(zB. Vivado: über 30 GB!)*, was in der gegebenen Kunden-Umgebung nicht möglich war.

Hier war zuweilen tiefergehende Analyse/Reverse-Engineering nötig, um einen vollautomatischen und hinreichend performanten Installationsprozess zu entwickeln.

- Xilix Vivado
- Sigasi Studio
- Sysgo PikeOS, Codeo

- MentorGraphics Modelsim, Questa
- Eclipse

Tools+Spachen: Debian, APT, Ansible, Vagrant, Jenkins, ESXI, Docker, Kubernetes, Vivado, Modelsim, Questa, PikeOS, Codeo, Eclipse, Sigasi

2.47 IOT-Lösung für Windkraftanlagen

- IOT-Lösung zur Umweltüberwachung an Windkraftanlagen
- Linux-Plattform / BSP (Debian-basiert)
- Portierung von Raspberry PI auf PC-Engines APU
- Treiber-Entwicklung für PC-Engines Mainboard + Peripherie
- Web-Interface und Cloud-Anbindung
- Remote-Update
- Build-Umgebung + Debian-Paketiereung + Deployment
- Testautomatisierung

Tools+Spachen: C/C++, GNU make, Linux-Kernel, Debian, Devuan, Jenkins, Git, Perl, Python, Shellscript, GNU/Linux

2.48 SW-Architektur: Verschlüsselte Mailboxes mit Mailbox-Sharing

- Verschlüsselte Speicherung von Mailboxen innerhalb einer Groupware
- Client-/Frontend-seitige Verschlüsselung *(kein Klaretext/Secrets auf dem Mailbox-Server)*
- Mehrstufiges Key-Management erlaubt Mailbox-Sharing und Vertretungsregelungen
- Anbindung von lokalen Key-Cards
- explizit HSM-freie Architektur *(Vermeidung eines SPOF/Bottleneck)*
- Zielgruppe: Rechtspflege

Tools+Spachen: Zimbra , Java, Python, SOAP, GnuPG, ZmPkg

2.49 Konzeption für Groupware-Integration des Elektronischen Gerichts- und Verwaltungspostfach (EGVP)

Der Endkunde ist eine öffentliche-rechtliche Körperschaft in der Rechtspflege, deren Mitglieder umfangreiche Dokumente mit den Gerichten mittels des Elektronischen Gerichts- und Verwaltungspostfach (EGVP) austauschen müssen.

Aktuell existieren hierfür nur rein Client-seitige Lösungen mit spartanischer Benutzerführung, aufwändigem Rollout *(nicht automatisierbar)* und Fehlen jeglicher Integrationsmöglichkeiten. Zudem sind diese *(entgegen offizieller Verlautbarungen)* ausschließlich für die grundsätzlich als unsicher einzustufende Windows-Plattform benutzbar.

Im Vorfeld des Projekts wurde dem Endkundem bereits eine Zimbra-basierte Groupware-Lösung geliefert, welche er seinen Mitgliedern *(und deren Mitarbeitern)* zur Verfügung stellt. In diese Groupware sollte EGVP derart integriert werden, daß es auch - ohne zusätzliche lokale Installationen - über die Web-GUI verwendet werden kann, ohne dabei die Ende-zu-Ende-Verschlüsselung zu brechen. Dabei sollen auch erweiterte Funktionen wie zB. Mailbox-Delegationen, Volltext-Suche, Termineinladungen, udgl. für EGVP nutzbar sein.

Desweiteren wurde hier eine Dokumentation der *(zu diesem Zeitpunkt noch undokumentierten)* Zimbra SOAP APIs für Provisionierung und Management aus externen Führungssystemen heraus geliefert.

Tools+Spachen: Zimbra , Java, EGVP, LaTeX, SOAP, GnuPG, ZmPkg

2.50 iOTP-/iTAN-Authentifizierung für Zimbra

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Für verschiedene Endkunden wird eine Multifaktor-Authentifizierung mittels iOTP *(aka. iTAN)* als Alternative zum existierenden OTP via SMS benötigt. Das Problem besteht hier darin, daß die SMS-Zustellung in verschiedenen Ländern nur sehr unzureichend funktioniert, sodaß hier oft die Kundenbetreuer angerufen werden müssen, um ein gültiges OTP zu erhalten. Dies sind idR. gleichzeitig auch Länder, in denen eine Postzustellung noch unzuverlässiger oder unsicherer als Telefonie/SMS ist.

Hierzu wurde als Alternative ein iOTP-Modus entwickelt, welcher kurz umrissen wie folgt agiert:

- Bei Problemen mit der SMS-Zustellung aktiviert der Kundenbetreuer den iOTP-Modus
- Es wird ein iOTP-Blatt generiert und dem Kunden innerhalb der Zimbra -Plattform zugestellt
- Dieser erfragt zu diesem Zeitpunkt noch das OTP telefonisch vom Kundenberater
- Nach Sicherung/Ausdruck des iOTP-Blatts aktiviert er dieses durch Eingabe eines iOTP
- Von nun an erfolgt die Authentifizierung mittels iOTP anstatt OTP
- Nach Verbrauch einer definerten Anzahl iOTPs wird ein neues Blatt generiert/gesendet

Die Implementation untergliedert sich in folgende Komponenten:

- Erweiterung der Middleware um iOTP-Datenstrukturen und -Workflows
- Rendering und Versenden des iOTP-Blatt als PDF mittels LaTeX

- Erweiterung der SAML-basierten SSO-Plattform zur iOTP-Abfrage
- Zimbra -Erweiterungen *(Zimlets)* zur iOTP-Verwaltung seitens Kunde und Kundenbetreuer

Tools+Spachen: Zimbra , Postfix, simplesamlphp, Jilter/Mailfilter, PHP, Java, Javascript, PostgreSQL , Shellscript, SAML, ZmPkg

2.51 Zimbra Mail History

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Verschiedene Kunden des Auftraggebers benötigten eine Lösung mittels derer innerhalb von Zimbra der Absender einer eMail nachvollziehen kann, an welche Empfänger diese gesendet und von wem diese wann gelesen wurde. *(ähnlich wie dieses zB. auch DE-Mail anbieten soll)*.

Die in diesem Projekt entwickelte Lösung zeichnet die einzelnen Mail-Ereignisse *(gesendet, empfangen, gelesen, etc)* und ordnet sie deart zu, daß der Absender den Zustellungs-/Lese-Status jederzeit bequem einsehen kann. Somit können beispielsweise bei Fristensachen die Zustellungen nachgewiesen werden.

Um einen reibungslosen Betrieb, sowohl in Cluster-Umgebungen *(Zimbra läßt sich schließlich auch in einem großen Cluster mit hundertausenden bis millionen Nutzern betreiben)*, aber auch mit externen Weiterleitungen, zu gewährleisten, waren tiefgreifendere Anpassungen an verschiedenen Fronten nötig. Hierzu mußte einiges an R&E-Arbeit geleistet und Zimbra in der Tiefe analysiert werden.

In diesem Projekt lag die Konzeption/Architektur und Entwicklungs-Leitung, sowie große Teile der Implementation in meiner Verantwortung. Die Entwicklung wurde von einem internationalen *(über verschiedene Kontinente verteilten)* Team geleistet.

Tools+Spachen: Zimbra , Java, Jilter/Mailfilter, Javascript, Mysql, Shellcsript, Postfix, ZmPkg

2.52 Middleware zum User-Management für Zimbra und andere Anwendungen

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Für verschiedenene Endkunden waren bereits tiefergehende Zimbra -Customizations geliefert worden. Mit verschiedenen Folgeprojekten wurden diese jedoch stetig komplexer und benötigten eigene Datenbasis, welche nicht mehr sinnvoll in Zimbra -eigene Datenstrukturen *(iW.*

LDAP) abzubilden waren. So wurde zB. für ein großes Bankhaus eine automatische Nutzer-Provisionierung *(einschließlich der Nutzerbeziehungen und Zugriffsberechtigungen)* aus einem Datafeed heraus implementiert.

Außerdem bestand die Problematik, für jedes Zimbra -Upgrade bei jedem Kunden die jeweiligen Customizations auf die entsprechenden Zimbra -Versionen zu portieren. Zwar war der Aufwand durch die zuvor von mir eingeführten SCM-Methodiken schon erheblich reduziert, dennoch verblieben hier nicht unerhebliche Aufwände pro Kunde und Zimbra -Version.

An dieser Stelle habe ich ein grundlegendes Redesign vorgenommen, welches jegliche kundenspezifische Logik und Datenbasis in einer Middleware kapselt. Damit wurden die zuvor jeweils kundenspezifischen Core-Customizations durch eine generische ersetzt, welche nun bei allen Kunden gleichermaßen eingesetzt werden kann.

Die Middleware ist in Java implementiert, benötigt jedoch keine JVM. Stattdessen wird hier ein direkt ausführbares ELF-Binary erzeugt, welches als klassischer Unix-Dämon gestartet wird. Dies reduziert den Resourcen-Verbrauch erheblich, aber vereinfacht auch Build, Deployment und Betrieb.

Zur Kommunikation zwischen Middleware und Satellitensystemen *(von denen Zimbra nun nur eines von mehreren ist)* wird einen einfaches HTTP-basiertest, REST-artiges Protokoll verwendet, welches Aufgrund seiner Einfachheit sehr einfach *(uA. auch in Shellscripts)* implementiert werden kann. Im Gegensatz zu klassischen Ansätzen wie zB. SOAP oder XMLRPC vereinfacht dies die Entwicklung erheblich und spart Resourcen.

Zur möglichst effizienten Code-Wiederverwendung werden im Entwicklungsverlauf stets eine generische und verschiedene kundenspezifische Branches gepflegt. Nach Möglichkeit werden neue Funktionalitäten oder zumindest die Schnittstellen generisch ausgelegt und in der generischen Branch implementiert, während die kundenspezifischen Branches - welche jeweils auf die aktuelle generische Branch rebase'ed werden und nur die rein kundenspezifischen Anpassungen enthalten. Dies gilt auch für das DB-Schema, in welchem generische und kundenspezifische Relationen in verschiedene Namensräume getrennt und fachspezifische Datenstrukturen mittels *(schreibbarer)* Views auf generische Abgebildet werden.

Die Implementierung erstreckt sich über folgende Komponenten:

- Middleware-Server *(klassicher Unix-Dämon, als .deb paketiert)*
- Datenbank-Instanz mit generischen und kundenspezifischen Schemata *(PostgreSQL)*
- kundenspezifischer Importer
- Provisionierer
- Mail-Filter zur Sicherung der Kommunikationskontrolle
- Zimbra Core-Customizations
- diverse Zimlets *(zB. middleware-basiertes Addressbuch)*
- SAML Login-Portal *(ggf. mit kundenspezifischen Modulen)*

Tools+Spachen: Zimbra , LDAP, simplesamlphp, Java, GCJ, Javascript, PostgreSQL , Shellscript, ZmPkg

2.53 Zimbra OpenERP Connector

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Um einen direkten Zugriff auf wichtige ERP-Funktionen *(OpenERP)* aus Zimbra heraus zu ermöglichen, wurde mit einem internationalen Team ein Connector-"Zimlet" entwickelt. Dieses ist einfach mittels *Zimbra Package Manager* zu installieren und kann sowohl auf OpenERP-6 als auch OpenERP-7 zugreifen.

Die Basis-Version ist als OpenSource erhältlich, eine kommerzielle Variante bietet darüber hinaus erweiterte Funktionalitäten.

Meine primäre Verantwortungsbereiche innerhalb des Projekts lagen in:

- Software-Architektur und technische Leitung
- Qualitäts-Management
- Release-Management
- Paketierung

Tools+Spachen: Zimbra , Java, Javascript, OpenERP, Python, ZmPkg

2.54 Zimbra Owncloud Client

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Um die in Zimbra zukünftig nicht mehr vorhandene Dateiablage zu ersetzen und diese auch zentral für andere Systeme verfügbar zu machen, wurde eine Owncloud-Erweiterung entwickelt. Diese liegt zwei Varianten, Einzelnutzer *(per User-Configuration)* und Enterprise *(zentrale LDAP-Provisionierung/Authentifizierung)* vor.

Die Basis-Version ist als OpenSource erhältlich, eine kommerzielle Variante bietet darüber hinaus erweiterte Funktionalitäten.

Meine primäre Verantwortungsbereiche innerhalb des Projekts lagen in:

- Software-Architektur und technische Leitung
- Qualitäts-Management
- Release-Management
- Paketierung

Tools+Spachen: Zimbra , Java, Javascript, ZmPkg , GNU make, OwnCloud

2.55 Zimbra Plone-Connector

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Um einen direkten Zugriff auf wichtige der Plone-Portalplattform *(welche der Kunde auch für sein eigenes Portal einsetzt)*, wie zB. schnellen Dokumenten-Upload, aus Zimbra heraus zu ermöglichen, wurde mit einem internationalen Team ein Connector-"Zimlet" entwickelt. Dieses ist einfach mittels *Zimbra Package Manager* zu installieren.

Die Basis-Version ist als OpenSource erhältlich, eine kommerzielle Variante bietet darüber hinaus erweiterte Funktionalitäten.

Meine primäre Verantwortungsbereiche innerhalb des Projekts lagen in:

- Software-Architektur und technische Leitung
- Qualitäts-Management
- Release-Management
- Paketierung

Tools+Spachen: Zimbra , Java, Javascript, Plone, Python, ZmPkg

2.56 Zimbra-basierte sicher Kommunikationsplatform zur Neukundenaquise für internationale Großbank

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Für eine internationale Großbank wurde bereits im Vorfeld sichere Plattform zur Kommunikation zwischen der Bank und ihren Kunden, sowie externen Anlageberatern entwickelt. In diese galt es nun auch, potentielle Neukunden - für die Dauer der Verhandlungen - aufzunehmen.

Die fachliche Szenario stellt sich wie folgt dar:

In Ländern mit schwacher Infrastruktur bzw. dünner Siedlungsstruktur unterhält das Bankhaus idR. keine oder nur wenige Niederlassungen. Um dennoch hier Neukunden, zB. für das Anlagegeschäft, aquirieren zu können, werden diese von externen Vertretern vor Ort an Kundenberater des Bankhauses vermittelt. Diese eröffnen temporäre Zugänge zur Zimbra -Plattform, ein Aktivierungscode wird dem Neukunden über einen separaten Kanal zugestellt. Mit diesem kann jener seinen Zugang aktivieren und zugleich seine bereits erfaßten Stammdaten bestätigen bzw.korrigieren. Anschließend steht ihm die Zimbra -Plattform zur Verfügung und die Verhandlungen mit dem Kundenberater können beginnen. Nach Abschluß der Verhandlungen wird dieser temporäre Zugang wieder geschlossen und die Kommunikation archiviert.

Zusätzlich kann der Kundenberater Zusatz-Informationen, zB. den Verhandlungs-Status verwalten. Sämtliche Aktionen *(zB. Zustandsänderungen, eMails, etc)* werden zudem für evtl. spätere Nachprüfungen in einem Journal erfaßt.

Die Implementation dieser Funktionalität gliedert sich in verschiedene Komponenten auf:

- Management-Tool für den Kundenberater *(Zimbra Erweiterung - Zimlet)*
- Middleware-Erweiterung: Daten und Zustandsautomat für die Neukunden-Daten
- Provisionier für die temporären Zimbra -Accounts
- Mail-Filter zur Journal-Erfassung der Mails
- Erweiterung der SSO-Plattform
- Self-Registration-Portal
- Report-Generierung als PDF mittels LaTeX

Tools+Spachen: LaTeX, Zimbra , Postfix, simplesamlphp, Jilter/Mailfilter, PHP, Java, Javascript, PostgreSQL , Shellscript, ZmPkg

2.57 Source-Control- und Release-Managent / Git-Migration / Contigous Integration

Der Auftraggeber bietet neben Systemintegration auch tiefergehendes Customizing von OpenSource-Anwendungen und Entwicklungen von Erweiterungen für selbige *(insbesondere Zimbra , OpenERP, Plone, OwnCloud)*.

Die einzelnen Entwicklungsteams hatten zunächst ihren Source-Code, aber auch *(meist sogar manuell erzeugte!)* Builds sehr unstrukturiert in einer Subversion-Repository abgelegt. Dabei mangelte es allerdings gänzlich an einer sauberen Repository-Organisation, strukturierten Entwicklungs-, Testing- und Abnahme-Prozessen, Versions-Schemata, reproduzierbaren Build-Prozessen, udgl. Letztlich wurde SVN hier nur als unstrukturierte Datenhalde *(Fileshare)* mißbraucht, einzelne Entwicklungsstände konnten nicht *(ohne Weiteres)* definiert herausgezogen und gebaut werden. Selbstredent hat dies imense unnötige Kosten und Verzögerungen verursacht und den Entwicklungsablauf großen Risiken ausgesetzt.

In meiner Rolle als Chief Architekt habe ich diese Mißstände in mehreren Schritten bereinigt:

- Git-Migration, Reproduktion der Historie

  Der Source-Code der einzelnen Produkte wurde an verschiedensten Stellen im SVN-Baum abgelegt, teils mit nicht zusammenhängenden Kopien unterschiedlicher Stände, partiell als ZIP-Archive, oft mit falschen Character-Sets, sowie auch vielen nicht zum Zugehörigen Dateien *(temporäre Files, Build-Outputs, fachfremende Files, etc)*

  Zur Bereinigung habe ich hier einzelnen Produkte jeweils in Git importiert und anschließend mit speziell entwickelten Filter-/Rewriting-Tools über die gesamte History hinweg unnötigen Ballast entfernt, falsche Codierungen korrigiert, Verzeichnis-Strukturen angepaßt und die History aus den einzelnen Fragmenten rekonstruiert.

Das Ergbnis war eine konsistente History, aus der die einzelnen änderungen *(diff's)* jederzeit sofort ersichtlch sind, sodaß nun einerseits definierte Versionsstände jederzeit reproduzierbar sind, aber auch ein detailiertes Change-Review möglich ist.

Desweiteren wurden die Entwicklungs-Teams im Umgang mit Git, sowie allgemeinen Source-Control-Methodiken trainiert.

- Repository-Management und Redmine-Integration

  Im Vorfeld hatten lediglich einige Entwickler Zugriff auf das SVN, andere haben Ihre Stände über lokale Fileshares abgeglichen. Projektspezifische Zugriffskontrolle gab es nicht. Selbstverständlich erzeugte dies enorme Zusatzaufwände, Verzögerungen und Fehlerquellen, welche sich als sehr teuer erwiesen.

  Als Lösung habe ich hier eine Redmine-Integration implementiert, welche automatisch zentrale Git-Repositories zu den einzelnen Redmine-Projekten anlegt und die Zugriffskontrolle anhand der jeweiligen Projekt-Rollen steuert. Somit war es nun möglich, ohne manuelle Zusatzaufwände, für die einzelnen Projekte Git-Repositories zur Verfügung zu stellen und je nach Bedarf Zugriffsrechte zuzuteilen.

- Branch-Policies, Versions-Schema, Release-Management

  Da im Vorfeld die Entwicklungs- und änderungsabläufe sehr unstrukturiert und schwer nachvollziehbar verliefen, habe ich für die Entwicklungsteams definierte Prozesse eingeführt.

  Zunächst habe Topic-Branches *(fein granulierte Tasks/Tickets und jeweils zugehörige Branches)* eingeführt, sodaß hier einzelne Aufgaben weitestgehend unabhängig voneinander abgearbeitet und deren Ergebnis vor Integration in den Hauptzweig begutachtet werden können. Dieser Ansatz steigerte die allgemeine Code-Qualität und verbesserte das tägliche Lernen im Job.

  Desweiteren mangelte es an einer klaren Versions-Systematik, welche es anderen Beteiligten *(Produktmanagement, Operating, etc)* nicht erlaubte, Rückschlüsse auf die Qualität des eines Versionssprungs *(zB. welche Konsequenzen und Querabhängigkeiten wären zu erwarten?)* zu ziehen. Deshalb habe ich hier ein striktes "Semantic Versioning" eingeführt, welches sich auch in der Branch-Struktur und Change-Management niederschlug.

- Reproduzierbare Build-Prozesse, Paketierung, Contigious Integration

  Der Entwicklung mangelte es gänzlich an definierten und automatisierten Build-Prozessen. Stattdessen wurden die Builds jeweils manuell vorgenommen, was nicht nur große Kosten und Verzögerungen, sondern auch erhebliche Risiken mit sich brachte. Ebenso mangelte es auch an automatisierbaren Deployment-Prozessen, was erhebliche Zusatzaufwände in Validierung und Operating nach sich zog.

  Zur Lösung des Dilemmas habe ich zunächst für die einzelnen Produkt- Bereiche automatisierbare Build-Systeme und Paketierung implementiert, sodaß jegliche Software direkt über das Paketmanagement der jeweiligen Zielplattform ausgerollt werden kann.

  Für Zimbra -Erweiterungen habe ich hier eigens das Tool ZmPkg entwickelt, welches, basierend auf der bewährten dpkg/Apt-Technologie basiert und innerhalb einer Zimbra-Umgebung eine eigenständige Paketmanagement-Infrastruktur zur Verfügung stellt.

Anschließend wurde Source-Control, Build, Paketierung mit einer Jenkins- basierten Contigious-Integration zusammengeführt, welche automatisch neue Versionsstände baut, paketiert *(die Pakete dabei in Repositories ablegt)*, auf Testsysteme ausrollt und Tests abfährt.

Durch diese Maßnahmen wurden die Aufwände im gesamten Entwicklungsablauf, von Design über Entwicklung, Testing bishin zum Rollout/Operating um *Größenordnungen (in einigen Teilen auf weniger als ein Drittel!)* reduziert.

Tools+Spachen: Zimbra , Java, Javascript, Git, Shellscript, ZmPkg , Release-Engineering, Jenkins, Dpkg, Apt

2.58 Zimbra-basierte Secure Communication Platform für internationale Großbank

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Für eine internationale Großbank wurde in mehreren Stufen eine sichere Plattform für die Kommunikation zwischen der Bank und ihren Kunden, sowie externen Anlageberatern entwickelt. Im Gegensatz zu gängigen Groupware-Umgebungen ist diese Plattform aufgrund der verstärkten Sicherheitsanforderungen scharf abgeriegelt:

- Mehrwege-Authentifizierung mittels RSA-Token
- Eingeschränkte Kommunikation: Nutzer können ausschließlich nach klar definierten Beziehungen miteinander kommunizieren *(zB. Kunden nur mit deren Beratern, die Berater nur innerhalb der Abteilung)* - jegliche davon abweichende Kommunikationsversuche werden technisch unterbunden.
- Limitierte Features: einige Standard-Features, die in diesem Szenario zu einem Sicherheitsleck werden könnten *(zB. herunterladen von eMails)* wurden stillgelegt.

Diese Anpassungen wurden mittels kundenspezifischer Core-Customizations gewährleistet, welche mittels *Zimbra Package Manager* paktiert und ausgerollt.

In einer weiteren Stufe wurden weitere Nutzertypen mit komplexeren Kommunikations-Beziehungen angebunden und diese direkt aus einer extern eingelieferten Datenquelle provisioniert.

Um den Entwicklungs- und Wartungsaufwand auch für die Zukunft gering zu halten *(schließlich müssen Zimbra -Customizations vor jedem Zimbra -Upgrade zunächst entsprechend auf die neue Version angepaßt werden)* haben wir eine generische Lösung über eine eigene Middleware implementiert, welche mittels einer einfachen REST-API angesprochen wird und auch die User-Provisionierung übernimmt. Die Zimbra -seitigen Erweiterungen und Core-Customizations sind derart generisch ausgelegt, daß sie lediglich fein granulierte Nutzer-Eigenschaften von der Middleware abfragt und damit Features, Berechtigungen, etc. steuert - diese werden als eigenständiges, standardisiertes Produkt *(ohne kundenspezfische Anpassungen)* angeboten. Kundenspezifische Anpassungen finden damit nur noch innerhalb der Middleware statt, womit sich die Customizing-Aufwände dramatisch reduzieren.

In diesem Projekt lag die Architektur-Planung, Entwicklungsleitung/Release-Management und Implementation der Middleware in meiner Verantwortung.

Tools+Spachen: Zimbra , Java, Javascript, PostgreSQL , Shellscript, ZmPkg

2.59 Zimbra CRM

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

Da viele Zimbra -Anwender eine leichtgewichtige, in Zimbra integrierte, CRM-Lösung benötigen, ohne dabei gleich zu einer großen Lösung, wie zB. OpenERP greifen zu müssen, wurde hierfür ein eigenes ”Zimlet” entwickelt, welches grundlegende CRM-Funktionen bereitstellt. Dieses ist einfach mittels *Zimbra Package Manager* zu installieren.

Die Basis-Version ist als OpenSource erhältlich, eine kommerzielle Variante bietet darüber hinaus erweiterte Funktionalitäten.

Meine primäre Verantwortungsbereiche innerhalb des Projekts lagen in:

- Software-Architektur und technische Leitung
- Qualitäts-Management
- Release-Management
- Paketierung

Tools+Spachen: Zimbra , Java, Javascript, ZmPkg , GNU make

2.60 Zimbra Package Management

Der Auftraggeber ist der führende Anbieter für Integration und Customizing der Enterprise-Groupware *Zimbra Collaboration Suite* in Europa, und entwickelt auch eigene Erweiterungen für Zimbra . Zu den Kunden gehören uA. internationale Großbanken, große Verbände und öffentliche Institutionen.

In dem Bereich ist es wichtig, daß Deployments und Upgrade von Zimbra -Erweiterungen und -Core-Customizations *(kundenspezifische Anpassungen in der Kernanwendung)* sicher, robust und automatisiert ablaufen, da sonst die Aufwände und Betriebsrisiken rasch kaufmännisch inretabel würden. Aber auch in der Entwicklung ist dies wichtig, um kontinuierlich konsistente Testumgebungen - ohne zusätzliche manuelle Aufwände herzustellen *(contigous integration, etc)*.

Mit den Zimbra -Bordmitteln läßt sich nur eine Art von Erweiterungen - die ”Zimlets” genannten GUI Erweitungen - automatisiert installieren und entfernen. Dies ist aber auch nur unter Einschränkungen möglich: bringt ein solches ”Zimlet” beispielsweise eigene (Java-)Bibliotheken mit, werden diese zwar installiert, aber können nicht mehr bei Deinstallations des Zimlets automatisch entfernt werden. Zudem ist keine Konflikt- Prüfung bzw. -Verhinderung möglich.

Für andere Arten von Zimbra -Erweiterungen, zB. den ”Server-Extensions” *(mit diesen lassen sich serverseitige Ereignisse, wie zB. neuer Maileingang oder diverse Nutzeraktionen verarbeiten)* oder auch Core-Customizations *(Anpassungen/Erweiterungen im Applikationskern)* bietet Zimbra von Haus aus keinerlei Deployment-Mechanismus.

Damit besteht das Problem, daß für jede *(nicht-triviale)* Zimbra -Erweiterung eigenens Installations-/Upgrade-/Deinstallations-Programme geschrieben werden müßten, und andererseits Administratoren nur schwer einen Überblick bekommen, welche Erweiterungen in welchen Versionen aktuell installiert sind. Dies macht das Management der Zimbra - Erweiterungen über den gesamten Lebenszyklus - von Entwicklung/Testing/Release-Management bishin zum Produktivsystem sehr aufwändig und teuer.

Um dieses Problem an der Wurzel zu lösen, habe ich dem Auftraggeber eine Paket-Management-System für Zimbra - ”ZMPKG” geliefert. Dieses basiert auf der bei professionellen GNU/Linux-Distributionen bewährte Paketmanagement-Technologie ”dpkg”+”apt”, agiert jedoch vom Betriebssystem unabhängig, nur innerhalb der Zimbra -Instanz. Es ist somit auch auf anderen GNU/Linux-Distributionen, nicht auf ”dpkg” basieren *(zB. RedHat oder SuSE)* einsetzbar - hier werden die noch benötigten Komponenten vom Installer mitgeliefert.

ZMPKG bildet den gesamten Prozess - beginnend vom Build, bishin zum Deployment, Upgrade und auch vollständiger Deinstallaion - ab. Der Administrator kann mit den gewohnten Werkzeugen nach Belieben Zimbra -Pakete aus dem zentralen Repositorium installieren und upgraden.

Parallel dazu wurde auch eine *Jenkins*-basierte automatisierte Build-/Test-Umgebung aufgesetzt, mit der von der Entwicklung kommende Änderungen automatisch übersetzt/paketiert und auf Testsysteme aufgespielt werden.

Mittels dieser Technologie wurden beim Auftraggeber die Aufwände für Entwicklung, Testing und Rollout rapide reduziert. Bisher aufwändige Deployment-Tasks wurden iW. auf einen Befehlsaufruf reduziert.

Die ZMPKG-Technologie steht als OpenSource-Produkt kostenfrei zur Verfügung.

Tools+Spachen: Zimbra , Shellscript, dpkg, apt, rpm, GNU make

3 R+D: Eigene Projekte, Produkte, Grundlagenforschung

3.1 Linux-Kernel: Modbus protocol stack

Ziel des Projekts ist die Entwicklung einer MODBUS Socket API im Linux-Kernel. Der Zugriff für Programme *(Userspace)* erfolgt *(analog zu CAN)* über die BSD Socket API *(AF_MODBUS)*.

Die Vorteile sind:

- Konsolidierte Entwicklung / Pflege der Treiber in der Kernel-Community
- Board-spezifische Konfiguration auch via Devicetree – unabhängig von der Anwendung – möglich.
- Einbindung von Modbus-Geräten über andere Kernel-Subsysteme, für generische / HW-unabhängige APIs, wie zB. Industrial-IO, Input Subsystem, LED-Subsystem, HW-Monitoring, uvm.
- Security: Firewall und Access Control via Netfilter
- Traffic-Analyse und Monitoring mittels GNU/Linux Standard-Tools
- Vermeidung unnötiger Context Switches *(zB. Realtime)*

Tools+Spachen: Linux-Kernel, Modbus, Socket-API, Kernel-Treiber, netfilter, Linux-Embedded

3.2 Modellierung und Code-Generator für CAN-Umgebungen

Bei der Entwicklung von CAN-basierten Systemen *(zB. Automotive)* entfällt in der Praxis ein recht großer Teil des Entwicklungsaufwands auf Spezifizierung und Implementation der reinen CAN-Kommunikation, zB. Definitiion der Signale/Nachrichten, Zuordnung der IDs *(nebst Sicherstellung der Kollisionsfreiheit und Priorisierung)* bishin zum Code fürs Protokoll-Handling in den einzelnen Steuergeräten.

Zur Erleichterung existieren bereits einige etablierte kommerzielle Produkte, die aber mit erheblichem Integrationsaufwand und Wartungsaufwand verbunden sind. Für einen konsequentes DevOps-Vorgehen mit möglichst hohem Automatisierungsgrad *(zB. Code-Generierung, Integration mit diversen Toolchains, Testbeds, Source control management, uvm.)* erfordern diese Tools noch erhebliche weitere Aufwände, welche rasch die Projekt- Kosten und -Dauer in die Höhe treiben.

Das OpenSource-Projekt ""CannedFood"" verfolgt hier einen grundsätzlich anderen Ansatz: es handelt sich nicht um ein Fertigprodukt, sondern einen Baukasten, der sich leicht integrieren und anpassen läßt, und ist auf die Denkweise des klassischen Software-Entwicklers und agile / verteilte Entwicklung ausgerichtet.

Die Configuration findet über leicht verständliche Text-Dateien *(YAML)* statt, was sowohl Erweiterungen, aber aber auch flexibles Source Control Management sehr einfach gestaltet. Dabei ist die Configuration selbst auch derart modular gestaltet, daß sich sowohl die Gesamtsicht, als auch Teilkomponenten *(zB. das gesamte Netzwerk vs. einzelne Steuergeräte)* leicht in derselben Datenquelle abbilden lassen, ohne dabei den einzelnen Teams größere Abhängigkeiten *(zB. Entwicklungsumgebungen, Beschränkungen auf bestimmte Betriebssysteme, etc)* aufzubürden. Je nach Projekt und Komponente können hier selektiv verschiedene Funktionalitäten genutzt oder auch neue hinzugefügt werden *(zB. Code-Generierung für konkrete MCUs, Prüfgeräte, etc)*.

Durch die Einfachheit und Anpaßbarkeit bei gleichzeitiger Verwendung einer gemeinsamen Configurations-Basis für alle Komponenten ergibt sich sowohl hohe Flexibilität als auch Vermeldung von Koordinationsaufwänden.

3.3 Plan9 subsystem für Linux

Plan9 ist ein an Unix angelehntes Forschungs-OS, welches große konzeptionelle Fortschritte bzgl. Komplexitätsreduktion, Sicherheit und Entwicklungs-/Wartungsaufwände - insbesondere für Verteilte Systeme gebracht hat.

Ziel des Projektes ist die nahtlose Integration eines Plan9-Subsystems in Linux, sodaß sich Plan9-Features direkt unter Linux nutzen lassen und ein Mischbetrieb von Linux- und Plan9-Anwendungen möglich ist.

Hierzu wird die existierende Virtualisierungs-/Container-Infrastruktur ausgebaut und Plan9-spezifische Features hinzugefügt, uA:

- Erweiterung der User-Namespaces auf textuelle Usernames
- Deaktivierung von SUID innerhalb User-/Mount Namespaces
- Mounting für unprivilegierte User
- caphash/capuse-Device für User-Wechsel
- Portierung des Factotum Authentifizierungs-Agent
- Anpassung der coreutils *(uA. login/su via Factotum)*
- sukzessive Implementation diverser Plan9-Services

Tools+Spachen: Linux-Kernel, Plan9, C, LXC, coreutils, gcc, make, bash, rc, factotum

3.4 Linux-Kernel: Text-basierte Kernel-Schnittstellen via Dateisystem

Die Interfaces zwischen Linux-Kernel und Userland sind derzeit meist binär und damit stark von der CPU-Architektur abhängig. So benutzt zB. **netfilter** ein auf netlink-Socket-basiertes Protokoll, das sowohl von Kernel als auch Userland codiert/decodiert werden muß.

Entwicklungen auf anderen Betriebssystemen, insbesondere Plan9, haben deutlich gezeigt, daß sich textbasierte Dateisysteme hervorragend als universelle Schnittstelle zwischen verschiedenen Prozessen und Kernel eignen. Der Datentransport wird erheblich vereinfacht, der Code viel schlanker und die Schnittstellen transparenter. Unter Plan9 ist praktisch jede Resource, bishin zu Netzwerk-Schnittstellen und CPUs netzwerktransparent via 9P-Dateisystem erreichbar.

Ziel des Projektes ist es, wichtige Kernel-Schnittstellen *(zB. netfilter,acpi, ipmi,...)* via synthetische Dateisysteme neu zu implementieren und damit die alten Binärschnittstellen abzulösen. Probleme mit Binär-Inkompatibilitäten gehören dann der Vergangenheit an, und die Schnittstellen sind über ein passendes Netzwerk-Dateisystem direkt Netzwerk-transparent.

Tools+Spachen: C/C++, GNU make, Linux-Kernel, GCC, 9P, Shellscript

3.5 GNU Droid – Android-Umgebung auf GNU/Linux

Android basiert auf dem Linux-Kernel sowie zahlreichen aus der GNU/Linux-Welt bekannten Paketen, unterscheidet sich jedoch gravierend von gängigen GNU/Linux-Distributionen.

Ziel des GNU-Droid-Projektes ist, diese beiden Welten sukzessive derart sukzessive zusammen zu führen, daß das Android-Laufzeitsystem *(vollständig oder auch nur dedizierte Komponenten)* auch in einem klassischen GNU/Linux-Desktop verwendet werden kann – um einerseits Android-Apps auch auf dem Desktop zu betreiben, andererseits Android-Komponenten *(Dalvik, GUI-System, etc)* für klassische Desktop-Anwendungen verwenden zu können.

Andererseits werden auch die Kernel-Zweige derart zusammen geführt, daß eine Android-Umgebung auch auf einem normalen vanilla-Kernel lauffähig ist, um so uA. Aufwände für die Portierung und Pflege auf spezifische Hardware dramatisch zu senken.

Perspektivisch sollen hier auch bewährte Konzepte und Technologien aus der klassischen GNU-Linux-Welt *(zB. Paketmanagement)* in die Android-Welt überführt und so Entwicklung, Pflege, Betrieb angepaßter Android-Systeme *(zB. anwendungsspezifischer Handgeräte)* deutlich vereinfacht werden.

Tools+Spachen: Android, Linux-Kernel, Git, C/C++, Java, Dalvik, cmake, GNU make, autotools, ptxdist, Shellscript

3.6 CleanChrome – Ballast-befreiter Chromium-Browser

Google's Chromium zählt zu den meist-verwendeten und funktionsreichsten Web-Browsern der heutigen Zeit. Jedoch ist dieser äußerst resourcenhungrig und wird mit zunehmend mehr und

mehr Misfeatures *(zB. CloudPrint, DRM, WebSQL, etc)*, übefrachtet, die schlicht nicht mehr Aufgabe eines Web-Browsers gehören und nur – sowohl bzgl. Resourcenverbrauch, als auch Wartungsaufwand unnötig kostspielig sind, und auch Gefahren für die Integrität und Sicherheit mit sich bringen.

Offenkundig fährt Google die Strategie, mit dem Browser zunehmend das Betriebssystem zu subsumieren und mehr und mehr dessen ureigener Aufgaben *(zB. Geräteverwaltung, insb. Drucker, Netzwerk, etc)* "in die Cloud" zu verlagern. Das ist nicht nur technisch grob unsinnig, sondern birgt auch die klare Tendenz, dem Nutzer die Hoheit über seine Daten, bzw. seines Computers an sich sukzessive zu entziehen *(nicht zuletzt beim Thema DRM – digital **restriction** management)*.

Um dieser gefährlichen Tendenz entgegen zu wirken, wird im CleanChrome-Projekt eine eingehende Verschlankung und Eliminierung von Misfeatures vorgenommen, zB.

- entfernen sämtlicher gebündelter externer Komponenten, welche stattdessen direkt aus der Distro verwendet werden
- Refactoring des Buildprozess für einfache Handhabung und Paketierung
- Modularisierung optionaler Features, sodaß diese auch als separate Pakete ausgerollt werden können.
- Eliminierung des CloudPrint-Misfeatures *derartige Funktionen – sofern überhaupt erforderlich – gehören in das Drucksystem des OS, wie zB. cups*
- Eliminierung des "Digital Restriction Management"-Misfeaures – Unterminierung des Eigentums am Gerät durch derartige Sabotagemaßahmen ist nicht hinnehmbar
- Eliminierung von Abhängigkeiten zu Desktop-Bus udgl.
- Umstellung des Video-Streaming auf GStreamer zur Nutzung von Hardware-Pipelines *zB. HW-Codecs und -Filter, CRTC-Overlays, etc*
- Auslagerung der Netzwerkzugriffe auf separate Daemons
- Auslagerung der Profildaten in offene und generische Formate
- ...

Tools+Spachen: Android, Git, C/C++, Java, Chromium, GNU make, autotools, ptxdist, Shellscript

3.7 OpenVZ Virtualisierung für Collax Business Server

Collax Business Server ist eine renommierte Linux-Appliance für Office-Server in der die KMU. Die Server-Appliance ist vollständig per Web bedienbar und kann für typische Office-Anwendungen einen Windows-Server ersetzen.

Aktuell mangelt es jedoch noch an der Unterstützung für virtuelle Server-Umgebungen. Diese wird mit dem Projekt auf Basis von OpenVZ nachgeliefert.

Die Collax-Appliance wird damit in die Lage versetzt, praktisch beliebig viele virtuelle Linux-Server zu beheimaten.

Tools+Spachen: C/C++, GNU make, Linux Kernel, CVS, Subversion, Javascript, Shellscript, Perl, Crosstool

3.8 AuctionWatch - Auktionssuchmaschine

Mit der Auktionssuchmaschine **AuctionWatch** erhält der Schnäppchenjäger ein einfach zu bedienendes Werkzeug zum regelmäßigen Stöbern nach interessanten Auktionen.

Beliebig viele Suchabfragen können definiert werden, die von der Suchmaschine regelmäßig abgerufen werden. Der Nutzer bekommt die jeweils neu gefundenen Artikel eingeliefert und kann sie ggf. sortieren lassen (zB. zu teure Artikel anhand eines Maximalpreises aussortieren). Einmal gelöschte Artikel erscheinen auch nicht nach erneuter Suche wieder.

Zum derzeitigen Stand kann nur auf eBay gesucht werden - Anbindungen an andere Auktionshäuser sind derzeit in Arbeit.

Tools+Spachen: C/C++, PHP, Java, Javascript, PostgreSQL , Apache, Subversion

3.9 Briegel Builder - vollautomatisiertes Buildsystem für Firmware und Distros

Als Embedded-Entwickler ist man oft mit dem Problem konfrontiert, fein kofektionierte Pakete und Firmware-Images für die unterschiedlichsten Zielsysteme zu erzeugen und dabei Seiteneffekte (zB. Verwicklungen mit dem Entwicklungssystem) auszuschließen.

Viele Entwickler benutzen dazu jeweils eigene Scripte für jedes Zielsystem. Das bringt aber schnell einen erheblichen Pflegeaufwand mit sich, besonders wenn sich Anforderungen/Spezifikationen öfters ändern.

Mit dem **Briegel Builder** wird dem Entwickler dieser Aufwand praktisch vollständig aus der Hand. Der Entwickler muß jedelich jedes Zielsystem einmalig configurieren (zB. Crosscompiling Toolchain erzeugen) und für jedes gewünschte Paket Version und Features wählen. Der restliche Prozeß wird von Briegel vollautomatisch übernommen, einschließlich Download, Patching und Abhängigkeiten bishin zur Paketierung.

Nicht alle Pakete sind im Original für feine Konfektionierung und sauberes Crosscompiling geeignet. Hierfür werden bereits fertige Patches mitgliefert.

Tools+Spachen: Java, PHP, GNU Make, Ant, PostgreSQL , Unitool, GNU Autotools, Kaffe, GCJ, Crosstool

3.10 Comprehensive Source Database

In der CSDB werden OpenSource-Pakete mit ihren Versionen und den dazugehörigen Download-URLs (der Quellpakete) erfaßt. Die Datenbank arbeitet mit streng normalisierten Versionsnummern und läßt sich automatisiert abfragen.

Somit ist eine Integration der CSDB in automatische Build-Systeme, die Quellpakete herunterladen müssen, einfach und unkompliziert möglich.

Der Datenbestand wird durch automatische Crawler, die uA. Websites und FTP-Server zahlreicher Opensource-Projekte absuchen, regelmäßig aktuell gehalten.

Tools+Spachen: C/C++, PHP, PostgreSQL , plsql, Perl, CVS, Shellscript

3.11 metux Elastic Grid - openVZ-basierte Cloud-Infrastruktur

Das Elastic Grid stellt eine virtualisierte Hosting-Plattform für Linux-basierte Anwendungen zur Verfügung. Damit lassen sich Linux-Server und -Appliances flexibel und effizient auf einen großen Cluster verteilen.

Im Gegensatz zu Projekten wie OpenNebula oder OpenQRM wird hier nicht auf Hardware-/Hypervisor-Ebene *(VMware,Xen,KVM,etc)* sondern mit Betriebssystem- Virtualisierung (OpenVZ) gearbeitet. Das zT. erheblich Resourcen und vereinfacht das Datenmanagement.

Ein einzigartiges Feature ist hier die direkte Verbindung der einzelnen VMs/VEs über 9P-Channels *(Plan9 IPC/network filesystem)*: die VEs bzw. darin laufendende Anwendungen kommunizieren untereinander über IO-Streams, die in einer vom Grid-Controller organisierten Dateisystemhierachie abgebildet werden. Das spart einerseits Vergabe/Management von IP-Adressen, Tunneln, Routing und ermöglicht andererseits eine sehr einfache Entwicklung von hochverteilten Anwendungen mit vollautomatischen Balancing/Failover.

Die Infrastruktur besteht vorallem aus den Bausteinen:

- **Host Node Facility:** Dient dem Hosting der einzelnen VE's. Das Deployment einzelner VEs erfolgt hier über Robot-Schnittstellen mit einem abgestuften Rechtemodell. So die einzelnen Schritte des Deployments *(Laden der Images, Mounting)* vollautomatisiert ferngesteuert werden.
- **Virtual Service Link Facility:** Verbindet die virtualisierten Kommunikationsendpunkte *(9P-Links)* zwischen den VEs, ebenso wie das Management von IP-Addressen/Routen und DNS.
- **Workload Management Facility:** Primäre Schnittstelle für die Allokation von VEs und anderen Resourcen, Upload der Images/Workloads, Accounting, sowie Verteilung der VEs auf HNFs nebst Configuration der VSLFs.

Als Workload sind praktisch alle auf OpenVZ-/LXC-Container angepaßte GNU/Linux- Distributionen geeignet. Es bietet sich jedoch an, für spezielle Anwendungen optimierte Images, zB. mittels *metux Briegel Builder* zu erzeugen.

Tools+Spachen: C/C++, GNU/Linux, OpenVZ, VPN, 9P, Java, Perl, PHP, OpenSSL/OpenSSH, Shellscript, PostgreSQL , Lighttpd

3.12 MediaCloud: Media-Portal Plattform

Das Produkt **MediaCloud** bietet die Basis-Infrastruktur für den Aufbau von größeren Medien-Portalen. Es wird hierfür in Portal-Systeme *(zB. Joomla)* eingebunden, die hierfür als Frontend dienen. Die MediaCloud-Infrastruktur übernimmt dabei Upload, Convertierung, Verwaltung und Wiedergabe der gesamten Mediendateien. An eine MediaCloud-Instanz können beliebig viele Portale mit individuellen Berechtigungen angebunden werden, sodaß beispielsweise Medien eines Portals in anderen sichtbar gemacht werden können.

Als Storage-Backend stehen hierfür Venti *(Single-node)* oder Nebulon Cloud *(Cluster)* zur Auswahl.

Tools+Spachen: C/C++, Plan9, Venti, PHP, Perl, PostgreSQL , Javascript, AJAX, Shellscript, Lighttpd, LAMP, Git, ffmpeg, mplayer, Flash, Red5

3.13 Nebulon Superstorage Cloud

Die Nebulon Storage Cloud ist ein Peer-2-Peer basierter, hochredundanter Cloud-Speicher für sehr große statische *(nicht veränderliche)* Datenmengen - bishin zur Petabyte-Skala. Hierbei werden die einzelnen Datenobjekte über ihren Content-Hash addressiert und automatisch auf dem Weg zum finalen Verbraucher repliziert.

Durch starke Verschlüsselung und entsprechende Datenstrukturen ist eine besondere Vertrauensstellung zu oder zwischen einzelnen Knoten nicht erforderlich - es können so freie Kapazitäten theoretisch allen im Netz verfügbaren Geräten - bishhin zu externen Endgeräten *(sogar Mobiltelefonen)* verwendet werden.

Auf dieser Basis lassen sich hochgradig verteilte und redundante Speicherplattformen *(zB. als Backend für Groupware-Lösungen, Content-Management, u.v.m, aber auch Cloudspeicher ähnlich Dropbox)* implementieren.

Darüber hinaus lassen sich statische Web-Inhalte *(durch Einsatz von Proxies und Browser-Erweiterungen)* am üblichen HTTP-Kanal vorbei, direkt via Nebulon übertragen und so dessen inherente Replikation anstelle von Caches nutzen. Hierzu meldet der Client dem Server mittels HTTP-Header seine Nebulon-Fähigkeit, und sodaß dieser zB. anstelle der Daten eine Weiterleitung auf das entsprechende Nebulon-Objekt senden, aber auch Links in HTML-Seiten ersetzen kann. Aber auch bei fehlender Client-Unterstützung kann diese Technologie innerhalb von Webserver-Clustern eingesetzt werden, analog zum klassischen Einsatz von Caching Reverse-Proxies *(squid, varnish, etc)*.

Tools+Spachen: C, C++, Java, GCJ, CNI, JNI, GNU/Linux, HTTP, Git

3.14 OpenZimbra - Community-Fork der Zimbra Collaboration Platform

Die **Zimbra** **Collaboration Suite** ist eine fortschrittliche Groupware- und Collaboration-Plattform für den Einsatz mittleren und großen Unternehmen, welche dank Cluster-Fähigkeiten gut auf

viele tausende Nutzer skaliert und leicht mittels verschiedenster Erweiterungen *(zB. Zimlets)* gut auf individuelle Bedürfnisse anpaßbar ist. Die Software liegt einerseits in einer quelloffenen "Community Edition" sowie einer kommerziellen "Network Edition" mit einigen Zusatzfunktionen *(zB. Outlook-Integration)* vor.

Um das Rollout der Erweiterungen zu vereinfachen, habe ich bereits vor geraumer Zeit ein Build- und Paketmanagement hierfür entwickelt, welches unterdessen sich in zahlreichen Kundenprojekten sehr bewährt hat *(siehe ZmPkg)*.

Jedoch haben sich in Zimbra einige Architektur-Schwächen herausgestellt, welche vom Upstream schon über Jahre nicht aufgegriffen werden. Zudem ist der Upstream teils sehr schwerfällig beim Bugfixing, und hat auch sonst kein sonderliches Interesse an der OpenSource-Community gezeigt.

Nach der kürzlichen Übernahme durch Telligent - ein Unternehmen, welches ursprünglich in ganz anderen Bereichen aktiv war und auch keine besondere Kompetenz im GNU/Linux- und Opensource-Bereich darstellen kann - kommen Zweifel an der Zukunft des Produktes auf. Dies wird erschwert durch die Änderung der Lizenzpolitik, nach welcher man den Integrations-Partnern den Einsatz der kostenfreien OpenSource-Edition untersagen und diese zur Umstellung aller ihrer Kunden auf die kommerzielle Version zwingen möchte.

Aus diesen Gründen habe ich mich entschlossen, einen eigenen Fork des Zimbra-Projekts zu starten und so den Grundstein für eine echte Community zu legen.

Die großen Meilensteine für den Fork stellen sich aktuell wie folgt dar:

- saubere SCM-Infrastruktur und Reparatur des Build-Systems
  (aktuell lassen sich die von Zimbra bereitgestellten Sourcen nicht out-of-the-box compilieren, es gilt hier noch zahlreiche Defekte und sogar fehlenden Code zu beheben)
- Bereinigung aller 3rd-Party-Komponenten
  (die aktuelle Zimbra-Codebase behinhaltet zahlreiche 3rd-Party-Komponenten in Binärform, welche stattdessen aus von der jeweiligen OS-Distribution gelieferten Paketen entnommen werden sollten)
- saubere Paketierung mittels entsprechender Distro-Technologien
  (aktuell werden die Distro-Pakete direkt mittels Lowlevel-Tools, an den üblichen Infrastrukturen, wie zB. pbuilder/buildd vorbei, direkt im Buildprozeß von Zimbra erzeugt. Stattdessen sollte dieser Vorgang umgekehrt und mit den üblichen Infrastrukturen der jeweiligen Distros abgewickelt werden)
- unabhängige Nachrüstung von NE-Features
  (die kommerzielle "Network Edition" enthält einige Features, wie zB. Outlook/ActiveSync-Integration, welche es mittels bereits verfügbarer OpenSource-Technologien nachzubilden gilt)

Das Projekt befindet sich aktuell noch in einem frühen *(experimentellem)* Stadium. Sponsoring oder Beteiligungen wären herzlich willkommen.

Tools+Spachen: C/C++, GNU make, Git, Java, J2EE, Jetty, Zimbra , ZmPkg , Perl

3.15 OpenSource Software QM Taskforce

Das OSS-QM Projekt hat sich zum Ziel gestellt, unabhängig von einzelnen Distributionen, Patches für zahlreiche Opensource-Pakete bereitszustellen. Diese umfassen einerseits rasche Bugfixes, andererseits Erweiterungen zur besseren Anpassung an individuelle Anforderungen *(zB. in Embedded-Umgebungen)*.

Aktuell pflegen die meisten Distros ihre eigenen Patches. Allzu oft sind diese jedoch nicht allgemeintauglich und können/werden nicht direkt in das Originalprojekt einfließen.

Im Zuge meiner vielfältigen Entwicklungsarbeit im Embedded-Umfeld mußten immer wieder Pakete angepaßt bzw. repariert werden. Meist sind es kleine oder eher ”formelle” Dinge, zB. sauberes Crosscompiling aus einer Sysroot-Umgebung, Installation in ein Prefix, saubere Imports, PkgConfig-Unterstützung, etc.

Um längerfristige Doppelarbeiten zu vermeiden habe ich die Anpassungen derart allgemein ausgelegt, daß sie für die meisten Distros direkt taugen und unmittelbar in die Originalprojekte einfließen können.

Die entstandene Patch-Repository ist so organisiert, daß für jede *(unterstützte)* Version der jeweiligen Pakete ein Patch automatsiert heruntergeladen und eingespielt werden kann. Man könnte sich also das OSS-QM-Projekt als eine Art Filter vorstellen, der theoretisch für jedes *(unterstützte)* Paket in allen wichtigen Versionen eine unmittelbar ”reparierte” Version bereitstellt, die sich aus dem Originalpaket plus dem zugehörigen OSS-QM-Patch zusammensetzt.

Das OSS-QM-Projekt wird von meinem Distro/Firmware-Toolkit *metux Briegel Builder* unmittelbar benutzt, sodaß dort keinerlei eigene Patches mehr vorkommen.

Tools+Spachen: Git, C, Java, Perl, GNU diffutils, Subversion, CVS, Git, PHP, GNU make, GCC, Crosstool, Shellscript