



Profil

Stefan Ruppert

27. Mai 2006

Copyright 2006 Ruppert-IT

Copyright ©2006 Ruppert-IT

Stefan Ruppert
Altkönigstrasse 7
65830 Krifel
Deutschland

Web: <http://www.ruppert-it.de/>
Mail: stefan@ruppert-it.de

Kein Teil dieses Dokuments darf ohne schriftliche Genehmigung des Autors in irgendeiner Form (Fotokopie, Digitalisierung oder ein anderes Verfahren) reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltsverzeichnis

1	Einleitung	2
1.1	Fachliche Schwerpunkte	2
1.2	Produkte	2
1.3	Projekt- und Teamarbeit	2
1.4	Patente	2
2	Leistungen	3
2.1	Web-Site Erstellung	3
2.2	Performance Monitoring	4
3	Persönliches	5
4	Projekte	6
4.1	Weiterentwicklung eines ATC-Systems zur Luftlagedarstellung . .	6
4.2	Standardisierung, Design und Implementierung des Application Response Measurement (ARM) Standards Version 4.0	9
4.3	Redesign und Implementierung einer hoch verteilten Display Soft- ware	11
4.4	Entwicklung eines CORBA-basierten Operation-Support-Systems	12
4.5	Portierung einer Unix C++-Anwendung (VADS) von OSF1/Alpha nach Linux/Intel	13
4.6	Qualitätssicherung einer C++-Anwendung unter Unix/X11	14
4.7	Performance Messungen von CORBA Anwendungen	15
4.8	Monitoring von verteilten CORBA Anwendungen	16
4.9	Ereignis-Filterung in einem verteiltem System (TCP/IP)	17
5	Diplomarbeit	18
6	Publikationen und Vorträge	19
7	Skills	21

1 Einleitung

1.1 Fachliche Schwerpunkte

Nach meinem Studium, das ich im Sommer 1997 erfolgreich an der Fachhochschule Wiesbaden, Fachbereich Informatik abgeschlossen habe, bin ich als freiberuflicher Informatiker tätig. Meine Tätigkeitsfelder umfassen vornehmlich Entwicklung und Beratung im Umfeld von Unix- und verteilten Systemen. Hierbei insbesondere Analyse und Design von prozeduralen und objektorientierten Software-Systemen, Entwicklung von Software in den Sprachen C, C++ und diverser Script-Sprachen, Performance-Analyse und Optimierung von C/C++ Anwendungen sowie Laufzeit und Performance-Analyse von verteilten Anwendungen mittels ARM (Application Response Measurement), Komponenten Technologie und Middleware Architekturen wie CORBA, sowie HyperText Verarbeitung.

1.2 Produkte

Als Mitbegründer der MyARM GbR kann ich das MyARM Produkt zur Performance Messung und Bewertung verteilter Transaktionen im C/C++ und Java Kontext auf Basis des Application Response Measurement (ARM) 4.0 Standards empfehlen. Nähere Informationen können sie unter <http://www.myarm.de/> finden.

1.3 Projekt- und Teamarbeit

Während meiner bisherigen Projektarbeit, habe ich sehr gerne meine Kenntnisse an Teamkollegen und Mitarbeiter weitergegeben und bin immer gerne bereit Fragen zu beantworten.

1.4 Patente

Da ein Großteil meiner Kenntnisse und angewandte Techniken aus dem Bereich von Open-Source und Freeware wie von der Free-Software-Foundation (FSF) entwickelt und gefördert, entstammen und ich diese für sehr wichtig erachte, werde ich keine Patente auf computerimplementierte Erfindungen, die Open-Source oder Freeware entgegenstehen, anmelden oder im Rahmen von Projekten Unterstützung für Patentanmeldungen für solche Erfindungen leisten.

2 Leistungen

2.1 Web-Site Erstellung

Nach gemeinsamer Ausarbeitung Ihrer Bedürfnisse erstellen wir Ihnen Ihre Web-Site nach aktuellem Stand der Technik. Das heisst:

- Browser unabhängig: Benutzung der Web-Site mit allen gängigen Browser (Firefox, Internet Explorer, Opera, Konqueror).
- W3C standardkonform (XHTML, CSS)
- Unabhängig von Bildschirmauflösungen. Kein Text wie *optimiert für 800x600 Bildpunkte*
- Bearbeitung von Grafiken, Logos und Fotos.

2.1.1 Referenzen

MyARM Gbr



Produkte für Performance Monitoring von verteilten Anwendungen.

Andreas Ruppert Dachdeckermeister GmbH



Dachdeckermeistergeschäft in Flörsheim.

2.2 Performance Monitoring

2.2.1 Performance Probleme?

Ihre (verteilte) Anwendung hat Performance Probleme und Sie wissen nicht genau warum? Wir bieten folgende generelle vorgehensweise zur Identifizierung und Lösung an oder fragen Sie uns nach einer individuelle Lösung.

1. Problemanalyse: Nach gemeinsamer Problembesprechung erstellen wir Ihnen ein Konzept zur Analyse Ihrer Performance Probleme.
2. Konzept für die Aufdeckung der Probleme. Mögliche Ansätze sind:
 - Geeignete Instrumentierung Ihrer Anwendung.
 - Instrumentierte Simulation von Benutzern.
 - Verwendung generischer Instrumentierungsumgebungen und Anwendungen.
3. Testläufe Ihrer Anwendung in einer Performance-Messumgebung und Aufzeichnung der relevanten Performance-Kenngrößen.
4. Analyse der gewonnenen Performance-Messdaten.
5. Verbesserung und Optimierung Ihrer Anwendung aufgrund der durchgeführten Performance-Analyse.
6. Gegebenenfalls Wiederholung/Verfeinerung der Analyse/Konzepte.

2.2.2 Messumgebung

Basierend auf dem Application Response Measurement (ARM) Standard der Open Group setzen wir die [MyARM](#) Messumgebung zur Messung von C/C++ und Java Anwendungen ein.

3 Persönliches

Folgende Tabelle listet die wichtigsten persönlichen Daten:

Name:	Stefan Ruppert
Geburtsdatum:	28. August 1969
Geburtsort:	Flörsheim / Main
Adresse:	Altkönigstrasse 7 65830 Kriftel
Telefon:	Festnetz: +49-6192/9616749 Mobil: +49-163/7339712
Telefax:	
EMail:	stefan@ruppert-it.de
Web:	http://www.ruppert-it.de/
Studium:	1991-1997 Informatik Fachhochschule Wiesbaden
Abschluß:	Informatik Diplom im Juni 1997 an der FH Wiesbaden
Titel:	Diplom-Informatiker (FH)
Sprachen:	Deutsch, Englisch
Honorar:	projekt- und ortsabhängig zwischen 70,- und 90,- EURO pro Stunde bei freier Mitarbeit
Verfügbar ab:	Juni 2006
Einsatzorte:	Deutschland, bevorzugt Rhein-Main-Gebiet und PLZ (5,6,7)
bevorzugte Arbeitsgebiete:	Betriebssysteme (Unix), verteilte Systeme (CORBA), Netzwerklösungen (TCP/IP), Performance Analyse und Optimierung (ARM)

4 Projekte

Die aktuellsten Projekte werden zuerst aufgeführt.

4.1 Weiterentwicklung eines ATC-Systems zur Luftlagedarstellung

Weiterentwicklung eines operationelles Air-Traffic-Control (ATC) Fallback-Systems zur Luftlagedarstellung (**DFS-Phoenix**). Rohe Radardaten (Plots) werden mittels eines sogenannten Trackers zu Flugspuren (Tracks) korreliert, diese mittels verschiedener Algorithmen weiterverarbeitet und angeschlossene sub- oder externe Systeme in ATC-spezifischen Datenformaten abgegeben. Innerhalb des Projekts wurde eine Anzeige-Software (Display, CWP - Controller Working Position) für Tracks und Kartenmaterial auf Qt 3 Basis implementiert.

Kurzbeschreibung Produktentwicklung:

- Entwicklung eines geographischen Filters mittels eines PointInPolygon Algorithmus in C und C++ (Qt). Verschiedenste Ausprägungen des Filters zur Filterung von Plots und Tracks innerhalb des Trackers.
- Entwurf und Entwicklung eines Redundanz Konzeptes für den Tracker-Prozeß (Master-Slave) inklusive des Datenabgleichs zwischen Master und Slave Tracker.
- Entwicklung eines Redundanz Konzeptes zur redundanten Übertragung von Daten mittels UDP-Broadcast bzw. UDP-Multicast.
- Erweiterung und Verbesserung des eigenen Encoder und Decoder (C) des Eurocontrol Asterix Standards. Der Eurocontrol Asterix Standard ist ein hochflexibles binäres Byte-Stream Format.
- Entwurf und Entwicklung eines modularen Dialog Konzeptes für die Display Komponente (mehr als 30 Dialoge).
- Entwicklung von verschiedenen Display Komponenten (Dialoge) für die CWP.
- Portierung der Software von Linux/x86 auf Linux/HP-PA.
- Entwicklung einer eigenen auf Qt 3 basierten C++ Logging Bibliothek ähnlich den log4j oder log4cpp Konzepten.
- Erarbeitung eines Konzeptes zur besseren Verwaltung von "Release Notes" in Bezug auf parallele Entwicklungszweige (CVS Branches).

Kurzbeschreibung administrative Aufgaben:

- Umstieg eines qmake (Qt) basierten Build-Prozesses auf GNU autotools.
- Installation und Etablierung eines Internet Web-Servers (Apache2) inklusive CVS Web-Frontend und Internet Wiki.
- Aufbau eines zentralen Repositories für die Entwicklungsumgebung, so dass Hinzufügen, Löschen oder einspielen einer neuen Version einfach und zentral möglich ist. Enthält alle benötigten Werkzeuge wie autotools, C/C++ Compiler, Werkzeuge zur Kode- und Dokumentationsgenerierung.
- Entwicklung von Skripten zur dauerhaften Qualitätskontrolle (Nightly builds, Unit Test Framework).
- Betreuung und Durchführung der Zusammenführung verschiedener Entwicklungszweige in den Hauptentwicklungszweig (CVS Mergen).
- Installation und Etablierung eines Bug-Tracking-Systems (Bugzilla)
- Analyse der existierenden C++ Software und Erarbeitung von Verbesserungen und Optimierungen für Laufzeitverhalten, Generalisierung (OOD) und Sicherheitsfragen wie Buffer-Overflows oder gefährlichen Codes.
- Konfiguration und Benutzung von QA-C++ zur Beurteilung und Verbesserung des C++ Quellcodes (statische C++ Quellcode Analyse).
- Mitwirkung und Unterstützung für Factory Acceptance Test (FAT) und System Acceptance Test (SAT).
- Unterstützung der Kollegen/Mitarbeiter in den Themengebieten C++ und Unix.

Projekt-Rolle:

Software-Architekt, Software-Entwickler, Qualitätssicherung Software.

Betriebssysteme:

Linux/x86, Linux/HP-PA

Entwicklungsumgebung:

GNU Make, GNU autotools, GNU C/C++-Compiler, CVS, Bugzilla, GNU Emacs, doxygen, Tiki Wiki, QA C++, gengetopt, gperf

Bibliotheken:

stdc++, Qt 3, CppUnit

Standards, Protokolle:

Eurocontrol Asterix, XML

Datenbanken:

SQLite

Hardware:

x86, HP-PA-RISC

Kunde, Branche:

Deutsche Flugsicherung GmbH, Verkehr

Zeit, Dauer:

Januar 2004 - Mai 2006

4.2 Standardisierung, Design und Implementierung des Application Response Measurement (ARM) Standards Version 4.0

Innerhalb der Open Group diskutierte die ARM Working Group, darunter vorallem die Firmen IBM, HP und tang-IT Consulting GmbH (für die ich innerhalb der Working Group tätig war), neue Anforderungen an den ARM-Standard. Der Standardisierungsprozeß für die Version 4.0 der Application Response Measurement (ARM) Spezifikation führte zu neuen APIs für die Programmiersprachen C und Java, die die neuen Anforderungen von Middleware- und verteilten Anwendungen an den ARM-Standard berücksichtigen.

Kurzbeschreibung Standardisierung:

- Auswertung und Umsetzung der Requirements von Anwendern und Implementieren von ARM an einen neuen ARM 4.0 Standard
- Zusammenführung der verschiedenen ARM 2.0 für C und ARM 3.0 für Java Standards in ARM 4.0, so dass es ARM 4.0 C Bindings und ARM 4.0 Java Bindings gibt, die einwandfrei zusammenarbeiten können
- Umsetzung der Requirements und Anpassungen von ARM 2.0/3.0 in C Funktionen und Strukturen und Java Schnittstellen für ARM 4.0
- Prototyping (Referenzimplementierung) des neu definierten ARM 4.0 für C APIs und Java APIs

Kurzbeschreibung Produktentwicklung:

- Entwurf einer modularen Komponenten-basierten ARM 4.0 Implementierung
- Implementierung des ARM-APIs in C (C++), möglichst Laufzeit optimal, um den Einfluß auf die zu messende Applikation so gering wie möglich zu halten
- Design und Implementierung von Backend- und Frontend-Modulen zur Datenablage, Datentransport und Datenanalyse
- Tools zur Datenanalyse (statistische Auswertungen, Transaktionskorrelation,...)
- Regression Test-Suite für den Basisteil der ARM Implementierung
- Erstellung der Web-Seiten für das tang-IT ARM Produkt (XHTML)

Projekt-Rolle:

Software-Architekt, Software-Entwickler

Betriebssysteme:

Linux/x86, Linux/PowerPC, Solaris/Sparc, TruUNIX/Alpha,
Windows 2000/x86

Entwicklungsumgebung:

GNU Make, GNU C/C++-Compiler, Intel ICC C++-Compiler, MS VisualC++
6.0, CVS, GNU Emacs, Java JDK 1.4, doxygen, L^AT_EX

Bibliotheken:

ACE, TAO, stdc++, PThreads, Sybase CT-Library, MySQL client Library

Standards, Protokolle:

ARM 2.0, ARM 3.0, ARM 4.0, XML, XHTML, CORBA

Datenbanken:

MySQL, Sybase

Hardware:

x86, PowerPC, Alpha, Sparc

Kunde, Branche:

The Open Group, Enterprise Management Forum, ARM (Application Re-
sponse Measurement) Working Group, Aktiv beteiligte Firmen: IBM/Tivoli,
HP, tang-IT Consulting GmbH.

Zeit, Dauer:

September 2001 - Dezember 2003

4.3 Redesign und Implementierung einer hoch verteilten Display Software

Redesign und Implementierung der internen, hoch verteilten, Display-Software-Architektur von DFS-VADS mittels verschiedener Design-Patterns. Design und Implementierung einer Schnittstelle zwischen der DFS internen Display-Software VADS und einer externen Flugplanungs- und Koordnungssoftware (FlipCof).

Kurzbeschreibung:

- Redesign der VADS-Software. Design-Patterns: View-Model-Controller, Singletons, OO generische Programmierung (templates)
- Unterstützung der Mitarbeiter in Fragen zu Unix, C++, CORBA, Performance tuning, Fehlersuche, Tracing und Debugging, Portierung auf Sun Solaris
- Anbindung einer externen Flugplanungs- und Koordnungssoftware (FlipCof) an die VADS interne Struktur.

Projekt-Rolle:

Entwickler, Berater und Software-Architekt

Betriebssysteme:

OSF1/Alpha, Linux/Intel, Solaris/Sparc

Entwicklungsumgebung:

GNU Make, GNU C/C++-Compiler, OSF1 CXX C++-Compiler, CVS, GNU Emacs, doxygen, Together++

Bibliotheken:

STL, ACE, TAO

Hardware:

PC x86, Alpha, Sparc

Kunde, Branche:

Deutsche Flugsicherung GmbH, Verkehr (Simulation)

Zeit, Dauer:

Dezember 2000 - Juni 2001, 7 MM

4.4 Entwicklung eines CORBA-basierten Operation-Support-Systems

Entwicklung eines Operation Support Systems (OSS) für IP-basierte Netzwerke auf Basis einer verteilten, multi-threaded Objekt-Architektur (CORBA)

Kurzbeschreibung:

- Aufbau des Build-Prozesses (GNU Make)
- (Re)design verschiedener Module (Objekte) und Komponenten innerhalb des OSS
- Implementierung der CORBA Schnittstellen der Module des OSS in C++ unter Verwendung des TAO ORBs.
- Unterstützung der Mitarbeiter in Fragen zu Unix, C++, CORBA, Performance tuning, Fehlersuche (memory-leaks (Insure), Tracing und Debugging)

Projekt-Rolle:

Entwickler, Berater und Software-Architekt

Betriebssysteme:

Linux 2.2, Solaris 2.7

Entwicklungsumgebung:

GNU Make, GNU C/C++-Compiler, GNU Debugger, CVS, GNU Emacs, doxygen, Together++, Insure

Bibliotheken:

STL, ACE, TAO

Hardware:

PC x86, Sparc

Kunde, Branche:

TRAIAN AG, Telekommunikation

Zeit, Dauer:

Juli 2000 - Oktober 2000, 4 MM

4.5 Portierung einer Unix C++-Anwendung (VADS) von OSF1/Alpha nach Linux/Intel

Kurzbeschreibung:

- Anpassung der Sourcecodes an den GNU C/C++-Compiler, inklusive Compiler-Padding Problematik (32bit vs. 64bit Architektur)
- Build-Prozeß für verschiedene Architekturen gleichzeitig aus einem Sourcetree
- Big- and Little-Endian Problemstellung

Projekt-Rolle:

Entwickler

Betriebssysteme:

DIGITAL Unix (OSF1), Linux

Entwicklungsumgebung:

OSF1 CXX C++-Compiler, GNU Make, GNU C/C++-Compiler, CVS, GNU Emacs, Doc++

Bibliotheken:

STL, X11/Motif 1.2, PThreads, ILOG Views, Generic++, GenericDisplay

Hardware:

PC Intel, DEC Alpha

Kunde, Branche:

Deutsche Flugsicherung GmbH, Verkehr (Simulation)

Zeit, Dauer:

März 2000 - Juni 2000, 2 MM

4.6 Qualitätssicherung einer C++-Anwendung unter Unix/X11

Qualitätssicherung und Optimierung einer C++-Anwendung unter Unix/X11 Window System (VADS, Very Advanced Display Software, Radar Display) (Im Bereich der Flugsicherungssimulation bei der Deutschen Flugsicherung (DFS)).

Kurzbeschreibung:

- Restrukturierung des gesamten Build-Prozesses, Anpassung der Sourcecodes an verschiedene Compiler Systeme, Dokumentation der Klassenhierarchie, Aufbau eines Intranet Web-Servers mit verschiedenen CGI-Skripten und Bereitstellung der erstellten Klassen Dokumentation mittels Doc++.
- Performance-Analyse der C++-Anwendung mittels Profiling Techniken (Speziell OSF1 Systemtool atom) und Optimierung der Anwendung mit den gewonnenen Ergebnissen.
- verschiedene Erweiterungen der Funktionalität der einzelnen Anwendung (X11/Motif, GenericDisplay, spezielle Radar-Display Funktionen).
- Unterstützung der Mitarbeiter in Fragen zu Unix und C/C++.

Projekt-Rolle:

Entwickler, Berater

Betriebssysteme:

DIGITAL Unix (OSF1)

Entwicklungsumgebung:

OSF1 CXX C++-Compiler, GNU Make, GNU C/C++-Compiler, CVS, GNU Emacs, Doc++

Bibliotheken:

STL, X11/Motif 1.2, ILOG Views, Generic++, GenericDisplay

Hardware:

DIGITAL Alpha

Kunde, Branche:

Deutsche Flugsicherung GmbH, Verkehr (Simulation)

Zeit, Dauer:

Dezember 98 - Dezember 99, 9 MM

4.7 Performance Messungen von CORBA Anwendungen

Entwurf und Implementierung von Performance Tools und Instrumentierung von CORBA-Servern (Inprise - VisiBroker) unter AIX für Performance-Messungen, sowie Instrumentierung von Java CORBA-Clients unter Windows NT (innerhalb eines Projektes bei IBM - Frankfurt für ein Telekommunikationsunternehmen).

Kurzbeschreibung:

- Entwurf und Implementierung der Instrumentierung einer CORBA-Anwendung (VisiBroker) mittels VisiBrokers Interceptor-Konzept.
- Entwurf und Implementierung zur interferenzarmen Antwortzeit Messung von CORBA Methodenaufrufen.
- Durchführung von Antwortzeit Messungen für standard CORBA Methodenaufrufe (VisiBroker) und deren statistische Auswertung.
- Anwendung von AIX Systemtools zum Monitoring der Systemperformance (SMP, Threads)

Projekt-Rolle:

Entwickler

Betriebssysteme:

AIX, Windows NT

Entwicklungsumgebung:

VisiBroker 3.2 C++/Java für AIX und Windows NT, AIX C Set C++ Compiler, Microsoft Visual C++-Compiler, IBM VisualAge for Java, GNU C/C++ Compiler, GNU Make, GNU Emacs

Hardware:

RS6000-Server, Intel-PC's

Kunde, Branche:

IBM Frankfurt, Telekommunikation

Zeit, Dauer:

Juli 98 - Oktober 98, 3 MM (*Im Oktober 98 wurde das gesamte Projekt eingestellt*)

4.8 Monitoring von verteilten CORBA Anwendungen

Monitoring von verteilten CORBA Anwendungen und deren Visualisierung. ObjectMonitor/VISCO. (Kooperation: Philips Forschungslaboratorien Aachen und Fachhochschule Wiesbaden, Fachbereich Informatik, Labor für verteilte Systeme).

Kurzbeschreibung:

- Entwurf und Implementierung der Instrumentierung einer CORBA-Anwendung (Orbix) mittels Orbix Request-Filterpunkte. (ObjectMonitor)
- Entwurf und Implementierung eines Ereignisprotokolls zwischen der Instrumentierung und der Visualisierung und deren Transport über eine TCP/IP socket Verbindung.
- Anpassung eines X11-Graph-Widgets an die Anforderungen der Visualisierung für CORBA-Anwendungen. (VISCO)
- Entwurf, Implementierung und Beschreibung der Schnittstellen für das Ereignisprotokoll, die die Visualisierungskomponente erwartet, so daß auch andere Sachverhalte visualisiert werden können.

Projekt-Rolle:

Entwickler, Software-Architekt

Betriebssysteme:

HP-UX, Solaris

Entwicklungsumgebung:

Orbix 1.3/2.0, Solaris C++ Compiler, HP-UX C++ Compiler, GNU C/C++ Compiler, GNU Make, GNU Emacs, GNU Debugger

Hardware:

HP-Workstation, Sun Sparc-Workstation

Kunde, Branche:

Drittmittelprojekt an der Fachhochschule Wiesbaden in Kooperation mit Philips Forschungslaboratorien Aachen, Forschungsgebiet Management von verteilten Anwendungen (CORBA)

Zeit, Dauer:

März 96 - August 97, 3 MM

4.9 Ereignis-Filterung in einem verteiltem System (TCP/IP)

Instrumentierung von C++-Anwendungen zur Erzeugung von Ereignissen (Methodenaufrufe, Attributänderungen) und deren Filterung und globale Weiterleitung über TCP/IP auf SunOS. (Kooperation: Gesellschaft für Mathematik und Datenverarbeitung (GMD) und Fachhochschule Wiesbaden).

Kurzbeschreibung:

- Entwurf und Implementierung von C++-Klassen zur Instrumentierung von C++-Anwendungen mit Hilfe des C++-Preprocessors mc4p, der von der GMD entwickelt wurde.
- Entwurf und Implementierung eines Konzepts zur Reduzierung der Interferenz zwischen der zubeobachtenden Anwendung und des Monitoring Systems mittels *Shared Memory Segmenten*.
- Entwurf und Implementierung eines TCP/IP-Netzwerk globalen Ereignissteuerungsprogramms, Global-Filter-Controller (GFC) genannt. Der GFC ermöglicht das Steuern der Ereignisse von instrumentierten C++-Anwendungen. Dabei kommuniziert der GFC mit Rechnerlokalen Ereignisfiltern über eine TCP/IP socket Verbindung.
- Entwurf, Implementierung und Beschreibung der Schnittstellen zur Einbindung in das Jewel++-Projekt der GMD.

Projekt-Rolle:

Entwickler, Software-Architekt

Betriebssysteme:

SunOS, HP-UX

Entwicklungsumgebung:

GNU C/C++-Compiler, GNU Make, GNU Emacs, GNU Debugger

Hardware:

Sun Sparc-Workstation, HP-Workstation

Kunde, Branche:

Drittmittelprojekt an der Fachhochschule Wiesbaden in Kooperation mit Gesellschaft für Mathematik und Datenverarbeitung (GMD), Forschungsgebiet Management von verteilten Anwendungen

Zeit, Dauer:

Okt. 1994 - Nov. 1995, 3 MM

5 Diplomarbeit

“Entwurf und Implementierung eines AmigaOS-Subsystems für den Mach-Mikrokern” (interne Diplomarbeit an der Fachhochschule Wiesbaden)

Kurzbeschreibung:

- Entwurf und Implementierung der AmigaOS Multitasking-, Speicher-verwaltung- und Interprozeßkommunikations-Schnittstellen und Subsysteme auf dem Mach-Mikrokern.
- Entwurf und Implementierung eines IDL-Compiler-Backends für die Generierung von C-Sourcecode von AmigaOS System-Object-Model, genannt Basic Object Oriented Programming Model for Amiga (BOOPA), Klassen.
- Entwurf und Implementierung von Speicher- und Interprozeßkommunikations- BOOPA-Klassen.
- Anpassen der AmigaOS disk operating Schnittstellen aus dem Free-ware AmigaOS Replacement Projekt (AROS).
- Erstellung einer Testsuite zur Überprüfung der Kompatibilität zur originalen AmigaOS-API.

Betriebssysteme:

OSF Mach 3.0, MkLinux, AmigaOS

Entwicklungsumgebung:

OSF Mach 3.0 DeveloperRelease 2.0, OSF MkLinux DeveloperRelease 2.0, GNU C/C++-Compiler, SUN IDL Compiler-Front-End, GNU Make, GNU Debugger (mit Mach Erweiterungen), GNU Emacs

Hardware:

PC x86

Zeit, Dauer:

Dezember 96 - Juni 97, 6 Monate

6 Publikationen und Vorträge

2004

- **“Application Response Measurement (ARM) Version 4.0 Software Development Kit (SDK)”**, Mitautor, *Hewlett-Packard Corporation (HP) und tang-IT Consulting GmbH (tang-IT) - Referenzimplementierung des ARM 4.0 Standards für die C und Java Programmiersprachen (2004)*.
Dokument und Software.

2003

- **“Application Response Measurement - Issue 4.0 - C Binding”**, Mitautor, *The Open Group - Technical Standard (2003)*.
Dokument.
- **“Application Response Measurement - Issue 4.0 - Java Binding”**, Mitautor, *The Open Group - Technical Standard (2003)*.
Dokument.

1998

- **“Using the AmigaOS-Datatypes-System”**, *Gateway Computer Show St. Louis Amiga’98, St. Louis, USA, 13.3.98*.
Vortrag.
- **“Developing an AmigaOS-Datatype”**, *Gateway Computer Show St. Louis Amiga’98, St. Louis, USA, 13.3.98*.
Vortrag.

1997

- **“Monitoring verteilter CORBA-Anwendungen”**, *Messen, Modellieren, Bewerten 97 (MMB’97), TU Freiberg, Sachsen, 17.-19.9.97*.
Vortrag und Short-Paper.
- **“Entwurf und Implementierung eines AmigaOS-Subsystems für den Mach-Mikrokern”**, *Diplomarbeit, intern FH-Wiesbaden, 17.6.97*.
Dokument und Vortrag.

1995

- **“Performance Management von verteilten Systemen”**, *Workshop Entwicklung und Management verteilter Anwendungssysteme (EMVA '95)*, Uni Dortmund, 9.-10.10.95
Short-Paper.

7 Skills

Betriebssystem-Erfahrung:

AIX (mittel), AmigaOS (sehr gut), TruUNIX 64 (DIGITAL Unix, OSF1) (gut), HP-UX (gut), Linux (sehr gut), Mach (gut), MacOS (wenig), MkLinux (gut), NetBSD (gut), Solaris (gut), SunOS (gut), Ultrix (gut), Unix allgemein (sehr gut), Windows NT (wenig), Windows 2000/XP (mittel)

Betriebssystem-Programmierung:

AIX (mittel), AmigaOS (sehr gut), TruUNIX 64 (DIGITAL Unix, OSF1) (gut), HP-UX (gut), Linux (sehr gut), Mach (gut), MkLinux (gut), NetBSD (gut), Solaris (gut), SunOS (gut), Ultrix (mittel), Unix allgemein (sehr gut), Windows (wenig)

Programmiersprachen:

C++ (sehr gut), C (sehr gut), Java (gut), Lisp (mittel), m68k Assembler (gut), Pascal (gut), Python (mittel), x86 Assembler (mittel)

Script-Sprachen:

csh (gut), Perl (wenig), REXX (gut), sh (gut)

Dokumentationssprachen:

AmigaGuide (sehr gut), HTML u. XHTML (sehr gut), L^AT_EX (gut), SGML u. XML (gut), TexInfo (gut)

Tools:

GNU Tools (sehr gut), CVS (gut), Doc++ (gut), doxygen (sehr gut), Insure (mittel), Subversion (sehr gut), Together++ (gut), QA C++ (gut)

Bug-Tracking:

Bugzilla (gut), Trac (sehr gut)

Datenbanken:

MySQL (gut), SQLite (gut), Sybase (mittel), Oracle (wenig)

Bibliotheken (C/C++):

ACE (mittel), ARM 4.0 (sehr gut), APR (Apache-Runtime) (gut), APR-util (Apache-Runtime util) (gut), Generic++ (gut), GenericDisplay (gut), ILOG Views (mittel), Motif (gut), MySQL client Library (gut), PThreads (gut), Qt3/Qt4 (gut), STL (gut), Sybase CT-Library (mittel), X11 (gut)

Standards:

CORBA 1.x/2.x: Common Object Request Broker Architecture (gut),
ARM 2.0/3.0/4.0: Application Response Measurement (sehr gut),
HTML4/XHTML: HyperText Markup Language (sehr gut),
SMTP: Simple Mail Transfer Protocol (gut),
HTTP: HyperText Transfer Protocol (Version 1.0/1.1) (gut),
CSS: Cascaded Style Sheets (gut),
C++ STL: C++ Standard Template Library (gut)

CORBA:

ORBs: Orbix (version 1.x) (gut), TAO (gut), VisiBroker (mittel)
Language-Mapping: C++ (gut), Java (wenig)
Komponenten: POA (mittel), Interceptors (gut), Services (wenig)

Methoden:

OOA (gut), OOD (UML) (gut), OOP (sehr gut)

Netzwerke:

TCP/IP (gut)

Interessengebiete:

Betriebssysteme (sehr gut), Hypertext-Systeme (sehr gut), Netzwerke (gut),
verteilte Systeme (sehr gut), Performance Analyse und Optimierung (sehr
gut)

Themengebiete:

Compilerbau (mittel), CORBA (gut), Hypertext (sehr gut), Netzwerkpro-
grammierung (TCP/IP) (gut), Systemprogrammierung (Unix) (sehr gut),
GNU Projekt (gut)

Projektverwaltung:

Wiki (mittel), Subversion (gut), CVS (sehr gut) und Trac (sehr gut)

Sonstiges:

Unix Administration (gut), WWW-, News- und FTP-Server Administration
(gut)

In Klammern angegeben, ist der Stand meiner Erfahrungen/Kenntnisse für den
jeweiligen Bereich. Dabei habe ich vier folgende Einstufungen vorgenommen:

sehr gut:

sehr gute Beherrschung, gute bis sehr gute Detailkenntnisse, absolut keine Einarbeitung erforderlich, Erfahrung in Anwendung und Entwicklung von über 3 Jahre.

gut:

gute Beherrschung, einige Detailkenntnisse, keine Einarbeitung nötig (evtl. in Details), Erfahrung in Anwendung und Entwicklung von über 1 Jahr.

mittel:

guter Umgang, keine Detailkenntnisse, evtl. erneute Einarbeitung nötig, Erfahrung in Anwendung und Entwicklung beschränkt sich auf 3-9 Monate.

wenig:

Erfahrungen am Rande verschiedener Projekt gesammelt, keine weitergehenden Detailkenntnisse, nur Grundkenntnisse, Erfahrung unter 3 Monate