



Profile

Stefan Ruppert

28th May 2006

Copyright 2006 Ruppert-IT

Copyright ©2006 Ruppert-IT

Stefan Ruppert  
Altkönigstrasse 7  
65830 Kriftel  
Germany

Web: <http://www.ruppert-it.de/>  
Mail: [stefan@ruppert-it.de](mailto:stefan@ruppert-it.de)

*No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior permission of the copyright owner.*

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Main focus . . . . .	2
1.2	Products . . . . .	2
1.3	Project and team work . . . . .	2
1.4	Patents . . . . .	2
<b>2</b>	<b>Services</b>	<b>3</b>
2.1	Web-Site creation . . . . .	3
2.2	Performance Monitoring . . . . .	4
<b>3</b>	<b>Personal</b>	<b>5</b>
<b>4</b>	<b>Projects</b>	<b>6</b>
4.1	Enhancements to an ATC system for air situation display . . . . .	6
4.2	Standardisation, Design and implementation of the Application Response Measurement (ARM) standards version 4.0 . . . . .	9
4.3	Redesign and implementation of a highly distributed display software . . . . .	11
4.4	Development of a CORBA based operation support system . . . . .	12
4.5	Port of an Unix C++-Application (VADS) from OSF1/Alpha to Linux/x86 . . . . .	13
4.6	Quality assurance of a C++ application under Unix/X11 . . . . .	14
4.7	Performance measurements of CORBA application . . . . .	15
4.8	Monitoring of distributed CORBA applications . . . . .	16
4.9	Event filtering in distributed systems (TCP/IP) . . . . .	17
<b>5</b>	<b>Thesis</b>	<b>18</b>
<b>6</b>	<b>Publications</b>	<b>19</b>
<b>7</b>	<b>Skills</b>	<b>21</b>

# **1 Introduction**

## **1.1 Main focus**

After finishing my computer-science study in summer 1997 at university for applied science Wiesbaden I worked as a freelancer in the following areas:

- Analysis and design of procedural and object-oriented software-systems.
- Distributed computing using CORBA and propriety protocols.
- Unix system programming.
- Performance analysis and optimisation of C/C++ applications.
- Performance analysis of distributed applications using ARM (Application Response Measurement).
- Standardisation of the Application Response Measurement - ARM 4.0.
- Hypertext processing (HTML, XML).

## **1.2 Products**

As co-founder of the MyARM GbR I can recommend the MyARM product family for measuring and estimation of response time performance of distributed transactions. MyARM supports the Application Response Measurement (ARM) 4.0 standard for C/C++ and Java. More information is available on our web-site <http://www.myarm.de/>.

## **1.3 Project and team work**

During my past projects I worked closely with team members and tried to share my knowledge with them.

## **1.4 Patents**

Most of my knowledge and used techniques results from work with Open-Source software above all the GNU tools provided by the Free Software Foundation (FSF). Therefore I will not support creating patents for computer implemented inventions if they contradict the open source philosophy!

## 2 Services

### 2.1 Web-Site creation

Together with you we elaborate your needs and develop your web site with state of the art techniques. This means:

- Browser independent: Web site can be used with all popular browsers (Firefox, Internet Explorer, Opera, Konqueror).
- W3C standard compliant (XHTML, CSS)
- Screen resolution independent. No phrases like *optimized for 800x600 pixel*.
- Processing of graphics, logos and photos.

#### 2.1.1 Referenzen

##### MyARM Gbr



Products for distributed application performance monitoring.

##### Andreas Ruppert Dachdeckermeister GmbH



Roofer company in Flörsheim, Germany.

## **2.2 Performance Monitoring**

### **2.2.1 Performance problems?**

Your (distributed) application has some performance bottlenecks and you do not know where they are? We offer the following general approach to identify the performance problems and its possible solution. Please feel free to ask for an individual offer.

1. Problem analysis: Together we discuss your performance problems. Based on this discussion we develop a concept for analysing the problem.
2. Possible concepts to discover the problems are:
  - Appropriate instrumentation of your application.
  - Simulation of users with instrumented applications.
  - Usage of generic instrumentation frameworks and environments.
3. Test deployment of your application with in a performance measurement environment and recording relevant performance metrics.
4. Analysis of recorded performance measurement data.
5. Improvement and optimisation of your application according to the results of the previous measurements.
6. Possible iteration and/or refinement of the analysis and concepts.

### **2.2.2 Measurement environment**

Based on the Application Response Measurement (ARM) standard of the Open Group we use the [MyARM](#) measurement environment for performance measurement of C/C++ and Java applications.

### 3 Personal

The following table lists the most important personal data:

<b>Name:</b>	Stefan Ruppert
<b>Birthday:</b>	28 August 1969
<b>Birthplace:</b>	Flörsheim / Main
<b>Address:</b>	Altkönigstrasse 7 65830 Kriftel Germany
<b>Phone:</b>	fixed: +49-6192/9616749 mobile: +49-163/7339712
<b>Telefax:</b>	
<b>Email:</b>	<a href="mailto:stefan@ruppert-it.de">stefan@ruppert-it.de</a>
<b>Web:</b>	<a href="http://www.ruppert-it.de/">http://www.ruppert-it.de/</a>
<b>University education:</b>	1991-1997 computer science at University of Applied Sciences Wiesbaden
<b>Graduation:</b>	Computer science diploma in June 1997 University of Applied Sciences Wiesbaden
<b>Title:</b>	Diplom-Informatiker (FH)
<b>Languages:</b>	German, English
<b>Charge:</b>	depends on the project and location, charge per hour between 70,- and 90,- Euro as a freelancer
<b>Available:</b>	June 2006
<b>Location:</b>	Germany, preferred Rhein-Main area (near Frankfurt/Main), Europe
<b>preferred work-scope:</b>	Operating systems (Unix), distributed systems (CORBA), net-solutions (TCP/IP), performance analysis and optimisation (ARM)

## 4 Projects

The most recent projects are listed first (from month.year-month.year):

### 4.1 Enhancements to an ATC system for air situation display

Further development of an operational air traffic control (ATC) systems for air situation display (**DFS-Phoenix**). Main radar data so-called plots are tracked by a so-called tracker to create a track for each air plane in flight. These tracks are transmitted to other sub- or external systems for further processing and/or just displaying the tracks on screen. For this purpose a ATC specific binary protocol (Eurocontrol Asterix) is widely used. Within the project a display software was developed to display tracks and map data based on the Qt 3 C++ framework.

#### **Brief description of product development:**

- Development of geographic filters using a point in polygon algorithm implemented in C and C++ (Qt). Different usages of this generic algorithm to filter plots and tracks within the tracker program.
- Design and implementation of a redundant master slave concept for the tracker process including communication between master and slave for data replication.
- Development of a concept for automatic master slave detection and negotiation between the tracker processes using UDP broadcast or multicast packets send by the tracker processes.
- Enhancement and improvement of the encoder and decoder for written in C of the Eurocontrol asterix standard. The Eurocontrol asterix standard is a high flexible byte oriented binary format.
- Design and implementation of a modular dialog concept (more than 30 dialogs) for the display component.
- Implementation of different dialogs for the display component of the controller working position (CWP).
- Porting the complete software from Linux/x86 to Linux/HP-PA.
- Implementation of a project logging framework based on the concepts of log4j or log4cpp.
- Design of a concept for managing “release notes” in a middle to large version control system (CVS) with many branches.

#### **Brief description of administrative tasks:**

- Changed build process of the complete software from qmake (Qt) to GNU based autoconf and automake tools.
- Installation and establishing an intranet web server (Apache2) used for serving a wiki and bug tracking for the project.
- Development of a central repository for automated builds of needed external software packages such as compiler, libraries and other tools. With this repository it is very simple to add or delete a tool for the projects development environment.
- Development of scripts for continuous quality assurance of the software (nightly builds, unit tests).
- Merging of different development branches into the main development branch of the software using version control features (CVS merging).
- Installation and establishing of a bug tracking system (Bugzilla).
- Analysing of the C/C++ software and development of improvements and optimisations for better code quality, runtime behaviour, more object oriented approaches and software security issues such as buffer overflows.
- Configuration and usage of a C++ source code analysis tool (QA-C++) to improve the C++ source code (static C++ source code analyse).
- Participation and support for factory acceptance test (FAT) and system acceptance test (SAT).
- Support for colleagues in topics to C/C++ and Unix.

**Project role:**

Software architect, Software developer, Software quality assurance.

**Operating systems:**

Linux/x86, Linux/HP-PA

**Development environment:**

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

**Libraries:**

stdc++, Qt 3, CppUnit

**Standards, Protocols:**

Eurocontrol Asterix, XML

**Databases:**

SQLite

**Hardware:**

x86, HP-PA-RISC

**Customer, Industry:**

German Air Traffic Control (Deutsche Flugsicherung GmbH), Transport industry

**Time, Duration:**

January 2004 - May 2006

## **4.2 Standardisation, Design and implementation of the Application Response Measurement (ARM) standards version 4.0**

Within The Open Group the ARM Working Group mainly driven by IBM, HP and tang-IT (for which I worked) discussed various aspects and requirements for the new version 4.0 for the Application Response Measurement standard. As the result of standardisation process of the new version 4.0 of the Application Response Measurement (ARM) specification new APIs for the programming languages C and Java were defined. These new APIs fulfils new requirements needed by middleware and distributed applications since versions 2.0 and 3.0 of the ARM standard.

### **Brief description:**

1. Standardisation:
  - Analysis and realisation of requirements from users and developers of ARM to the new 4.0 standard.
  - Joining the different standard version 2.0 for C and 3.0 for Java to a new compatible version 4.0 for C and Java.
  - Implementation of the requirements for ARM 4.0 into C and Java interfaces.
  - Prototyping (reference implementation) for the new 4.0 C and Java APIs.
2. Product development:
  - Design and implementation of a modular component based ARM 4.0 agent implementation.
  - Implementation of the ARM-APIs in C (C++), run time optimised to reduce interference with the instrumented application to a minimum.
  - Design and implementation of backend- and frontend components to store and retrieve measured ARM data.
  - Tools for analysing ARM data (statistical analysis, transaction correlation,...)
  - Implementation of a regression test suite of the core ARM 4.0 agent part.
  - Writing web pages for the tang-IT ARM. product (XHTML).

**Project role:**

Software architect, Software developer

**Operating systems:**

Linux/x86, Linux/PowerPC, Solaris/Sparc, Tru64 Unix/Alpha,  
Windows 2000/x86

**Development environment:**

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

**Libraries:**

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

**Standards, Protocols:**

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

**Databases:**

MySQL, Sybase

**Hardware:**

x86, PowerPC, Alpha, Sparc

**Customer, Industry:**

The Open Group, Enterprise Management Forum, ARM (Application Response Measurement) Working Group, Active involved companies: IBM/Tivoli, HP, tang-IT Consulting GmbH.

**Time, Duration:**

September 2001 - December 2003, 23 MM

### **4.3 Redesign and implementation of a highly distributed display software**

Redesign and implementation of the highly distributed display software (VADS) architecture at the German air traffic control using various design patterns. Design and implementation of an interface between the display software and an external flight planing and coordination software.

#### **Brief description:**

- Redesign of the VADS-Software using design patterns: View-Model-Controller, Singletons, OO template programming.
- Assist the employees in issues about Unix, C++, CORBA, performance tuning, bug tracking, tracing and debugging, porting to Solaris.
- Connecting the VADS software to an external flight planing and coordination software.

#### **Operating systems:**

OSF1/Alpha, Linux/x86, Solaris/Sparc

#### **Development environment:**

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

#### **Libraries:**

STL, ACE, TAO

#### **Hardware:**

x86, Alpha, Sparc

#### **Customer, Industry:**

German Air Traffic Control (DFS), Transport industry

#### **Date, Duration:**

December 2000 - June 2001, 7 MM

## **4.4 Development of a CORBA based operation support system**

Development of a CORBA based operation support system for TCP/IP networks based on a distributed, multi-threaded object architecture (CORBA).

### **Brief description:**

- Development of the build process (GNU Make)
- (Re)design of different modules (objects) and components of the OSS
- Implementation of the CORBA interfaces using C++ and the TAO ORB.
- Support of employees in issues about Unix, C++, CORBA, Performance tuning, bug tracking (memory-leaks (Insure), tracing and debugging)

### **Operating systems:**

Linux 2.2, Solaris 2.7

### **Development environment:**

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

### **Libraries:**

STL, ACE, TAO

### **Hardware:**

x86, Sparc

### **Date, Duration:**

July 2000 - October 2000, 4 MM

## **4.5 Port of an Unix C++-Application (VADS) from OSF1/Alpha to Linux/x86**

### **Brief description:**

- Adapt source codes to GNU C/C++-Compiler including padding problem (32-bit versus 64-bit system)
- Development of the build process (GNU Make) for different architectures simultaneously
- Big and little endianness difficulty

### **Operating systems:**

DIGITAL Unix (OSF1), Linux

### **Development environment:**

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

### **Libraries:**

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

### **Hardware:**

x86, Alpha

### **Customer, Industry:**

German Air Traffic Control (DFS), Transport industry

### **Date, Duration:**

March 2000 - June 2000, 4 MM

## 4.6 Quality assurance of a C++ application under Unix/X11

Quality assurance and optimisation of a C++ application under Unix/X11 Window System (VADS, Very Advanced Display Software, Radar Display) (In the air traffic control simulation area at german air traffic control (DFS)).

### **Brief description:**

- Restructuring the whole build process, adapt source code for different compilers, documentation of class hierarchy, establishing an intranet web-server with different CGI scripts for serving the documentation.
- Performance analysis of the C++ application with profiling techniques (used special OSF1 system tool atom) and optimisation of the application regarding the results of the analysis.
- different enhancements of the application functionality. (X11/Motif, GenericDisplay, special Radar-Display functions).
- Support for colleagues in topics to C/C++ and Unix.

### **Project role:**

Software developer, Consultant

### **Operating systems:**

DIGITAL Unix (OSF1)

### **Development environment:**

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

### **Libraries:**

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

### **Hardware:**

DIGITAL Alpha

### **Customer, Industry:**

German Air Traffic Control (Deutsche Flugsicherung GmbH), Transport industry.

### **Time, Duration:**

December 98 - December 99, 9 MM

## **4.7 Performance measurements of CORBA application**

Design and implementation of performance measurement tools for CORBA applications. Instrumentation of CORBA servers (Inprise - VisiBroker) running under AIX operating system and of Java CORBA clients running under Windows NT for performance measurements (within a project by IBM - Frankfurt for a telecommunication company).

### **Brief description:**

- Design and implementation of CORBA (VisiBroker) application instrumentation using VisiBrokers interceptor concept.
- Design and implementation of response time measuring for CORBA methods with less interference as possible.
- Execution of response time measurements of standard CORBA method invocations (VisiBroker) and their statistically analysis.
- Deployment of standard AIX system tools for system performance monitoring (SMP, Threads).

### **Project role:**

Software developer

### **Operation systems:**

AIX, Windows NT

### **Development environment:**

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

### **Customer, Industry:**

IBM Frankfurt, Telecommunication

### **Time, Duration:**

July 98 - October 98, 3 MM (*In October 98 the whole project was cancelled.*)

## 4.8 Monitoring of distributed CORBA applications

Monitoring of distributed CORBA applications and their visualisation. Object-Monitor/VISCO. (Cooperation: Philips research laboratories Aachen Germany and University of Applied Science Wiesbaden, Faculty computer science, Labour for distributed systems).

### **Brief description:**

- Design and implementation of CORBA application instrumentation (Orbix) with Orbix request filtering. (ObjectMonitor)
- Design and implementation of a protocol to exchange informations between the instrumented application and the display program using a TCP/IP socket transport layer.
- Adaptation of a X11 graph widget to the requirements needed to visualise the relationships of measured CORBA applications. (VISCO)
- Design, implementation and description of the protocol interface so that the visualisation component can be used in other context.

### **Project role:**

Software developer, Software architect

### **Operating systems:**

HP-UX, Solaris

### **Development environment:**

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

### **Customer, Industry:**

External project at University of Applied Science Wiesbaden in cooperation with Philips research laboratories Aachen Germany, Research area Management of distributed applications (CORBA).

### **Time, Duration:**

March 96 - August 97, 3 MM

## 4.9 Event filtering in distributed systems (TCP/IP)

Instrumentation of C++ applications for creating events of method invocations and attribute changes. Filtering and transportation of generated events in a distributed environment using TCP/IP protocol under SunOS. (Cooperation: Organisation for mathematics and data processing (Gesellschaft für Mathematik und Datenverarbeitung (GMD)) and University of Applied Science Wiesbaden).

### **Brief description:**

- Design and implementation of C++ classes for instrumentation of C++ applications using the C++ preprocessor mc4p. A tool from GMD.
- Design and implementation of a concept to reduce the interference of the instrumented application and the monitoring system using *shared memory segments*.
- Design and implementation of a network global event control application called global filter controller (GFC). The GFC application enables the control of instrumented C++ applications from a central point using TCP/IP connections to local event filters.
- Design, implementation and description of the interface for integration into the Jewel++ project at GMD.

### **Project role:**

Software developer, Software architect

### **Operating systems:**

SunOS, HP-UX

### **Development environment:**

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

### **Hardware:**

Sun Sparc-Workstation, HP-Workstation

### **Customer, Industry:**

External project at University of Applied Science Wiesbaden in cooperation with Organisation for mathematics and data processing (Gesellschaft für Mathematik und Datenverarbeitung (GMD) Research area: Management of distributed applications.

### **Time, Duration:**

October 1994 - November 1995, 3 MM

## 5 Thesis

“Design and implementation of a AmigaOS subsystem for the Mach microkernel”  
(internal thesis at the University of Applied Sciences Wiesbaden).

### **Brief description:**

- Design and implementation of the AmigaOS multitasking-, memory-management- and interprocess communication-subsystems on top of the Mach microkernel.
- Design and implementation of a IDL compiler backend for generating C source code of AmigaOS System-Object-Model classes. So called Basic Object Oriented Programming for Amiga (BOOPA).
- Design and implementation of memory and IPC BOOPA classes.
- Porting the AmigaOS disk operating interfaces of the open-source AmigaOS Research Project (AROS)
- Creating a test suite for checking the compatibility to the original AmigaOS API.

### **Operating systems:**

OSF Mach 3.0, MkLinux, AmigaOS

### **Development environment:**

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 (with Mach extensions), GNU Emacs

### **Hardware:**

PC x86

### **Date, duration:**

December 96 - June 97, 6 months

## 6 Publications

2004

- **“Application Response Measurement (ARM) Version 4.0 Software Development Kit (SDK)”**, co-author, *Hewlett-Packard Corporation (HP) and tang-IT Consulting GmbH (tang-IT) - Reference implementation of the ARM 4.0 standard for the C and Java programming languages (2004)*.  
Document and software.

2003

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

1998

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

1997

- **“Monitoring distributed CORBA applications”** (in german), *Measuring, modelling, rating 97 (MMB’97), TU Freiberg, Sachsen, 17.-19.9.97*.  
talk and short paper
- **“Design and implementation of a AmigaOS subsystem for the Mach microkernel”** (in german), *Thesis, intern university of applied science Wiesbaden, 17.6.97*.  
talk and full paper

1995

- **“Performance management of distributed systems”** (in german),  
*Workshop development and management distributed application systems (EMVA '95), Uni Dortmund, 9.-10.10.95.*  
short paper

## 7 Skills

### OS experience:

AIX (fair), AmigaOS (very good), TruUNIX 64 (DIGITAL Unix, OSF1) (good), HP-UX (good), Linux (very good), Mach (good), MacOS (marginal), MkLinux (good), NetBSD (good), Solaris (good), SunOS (good), Ultrix (good), Unix in general (very good), Windows NT (marginal), Windows 2000/XP (fair)

### OS programming:

AIX (fair), AmigaOS (very good), TruUNIX 64 (DIGITAL Unix, OSF1) (good), HP-UX (good), Linux (very good), Mach (good), MkLinux (good), NetBSD (good), Solaris (good), SunOS (good), Ultrix (fair), Unix in general (very good), Windows (marginal)

### Programming languages:

C++ (very good), C (very good), Java (fair), Lisp (fair), m68k Assembler (good), Pascal (good), Python (fair), x86 Assembler (fair)

### Script languages:

csh (good), Perl (marginal), Rexx (good), sh (good)

### Documentation languages:

AmigaGuide (very good), HTML u. XHTML (very good), L<sup>A</sup>T<sub>E</sub>X (good), SGML u. XML (good), TexInfo (good)

### Tools:

GNU Tools (very good), CVS (good), Doc++ (good), doxygen (very good), Insure (fair), Subversion (very good), Together++ (good), QA C++ (good)

### Bug-Tracking:

Bugzilla (good), Trac (very good)

### Databases:

MySQL (good), SQLite (good), Sybase (fair), Oracle (marginal)

### Libraries (C/C++):

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

### Standards:

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

**CORBA:**

**ORBs:** Orbix (version 1.x) (good), TAO (good), VisiBroker (fair)  
**Language mapping:** C++ (good), Java (marginal)  
**Components:** POA (fair), Interceptors (good), Services (marginal)

**Methods:**

OOA (good), OOD (UML) (good), OOP (very good)

**Networks:**

TCP/IP (good)

**Areas of interest:**

Operating systems (very good), Hypertext and document processing (very good), Networking (good), distributed systems (very good), Performance analysis and optimisation(very good)

**Topics:**

Compiler construction (fair), CORBA (good), Hypertext (very good), Network programming (TCP/IP) (good), System programming (Unix) (very good), GNU project (good)

**Project management:**

Wiki (fair), Subversion (good), CVS (very good) and Trac (very good)

**Miscellaneous:**

Unix administration (good), WWW-, News- and FTP-Server administration (good)

I assessed my knowledge and stated a grade for each area. Here is a short explanation / definition of the grading:

**very good:**

very good handling, detail knowledge good or better, absolutely no training needed, experience in development and deployment over 3 years.

**good:**

good handling, little detail knowledge, no training needed (perhaps in some details), experience in development and deployment over 1 year.

**fair:**

good handling, no detail knowledge, perhaps some new orientation needed, experience in development and deployment only about 3-9 months.

**marginal:**

marginal experience only, no detail knowledge, only basic knowledge experience less than 3 months.