

**Profile**  
Sep 2011

name	Menno RUBINGH
profession	<b>R&amp;D Software Designer, R&amp;D Software Documentation Writer</b>
contact info	postal address Jansonstrasse 18, 07745 Jena, Germany email <a href="mailto:hgnibur@freenet.de">hgnibur@freenet.de</a> phone (land line) +49 . 3641 . 217678 phone (mobile) +49 . 174 . 8215421
date of birth	7 Dec 1966
nationality	Netherlands
languages	English -- fluent German -- good Dutch -- native language French, Italian, Danish -- understanding (Japanese, Chinese -- some basic knowledge)
personality type	Big Five : O++ C- E-- A- N= MBTI : <a href="#">INTP</a>
Internet	Profile English (HTML) <a href="http://www.rubinghscience.org/cv/profile_en.html">http://www.rubinghscience.org/cv/profile_en.html</a> Profile English (PDF) <a href="http://www.rubinghscience.org/cv/profile_en.pdf">http://www.rubinghscience.org/cv/profile_en.pdf</a> Profile German (HTML) <a href="http://www.rubinghscience.org/cv/profil_d.html">http://www.rubinghscience.org/cv/profil_d.html</a> Profile German (PDF) <a href="http://www.rubinghscience.org/cv/profil_d.pdf">http://www.rubinghscience.org/cv/profil_d.pdf</a> Reference letters <a href="http://www.rubinghscience.org/cv/bew_unterl.html">http://www.rubinghscience.org/cv/bew_unterl.html</a>

**SUMMARY:**

I'm useful everywhere where new things need to be understood, then engineered into working prototypes and/or documented clearly. I'm good at the creative process of moving from a vaguely formulated problem to a working technical solution, and at the process of providing clarity into complex algorithms and/or source code.

**SERVICES OFFERED:**

- **Software structure & algorithm design --**
  - Ferreting out new possibilities and following these up to the creation of working prototype systems
  - Engineering the concrete methods and algorithms needed for the usage possibilities envisaged by the client (incl. numerical and AI algorithms)
  - Improvement of the structure and organization of existing code, to make it more maintainable
  - Design of software components that are: platform-independent, maximally bug-free, and clearly (understandably) structured
  - Creation of software tools.
- **Software documentation --**

Writing texts that

  - Convey insight into the underlying logic and mechanisms used in a piece of software
  - Give an overview over the internal design of a piece of software
  - Link up the logical overview picture with the actual software code.

**TECHNICAL EXPERIENCE DETAILS:**

**Programming Languages** ANSI C, C++, Java, Pascal, Perl, assembler, FORTRAN (-77 and -90), Lisp, Smalltalk, BASIC, VBA, UNIX shell scripts (sh/bash/ksh/C), Javascript, ...

**Operating systems and Platforms** UNIX (Linux, HP-UX, AIX), Windows, Renesas M32192 with OSEK/VDX, OS/2, DOS, VAX-VMS, Apple Macintosh, ...

**Software tools** gcc, gdb, ddd, gprof, make, Microsoft Visual C++, Lauterbach debugger, Vector tools, yacc/bison, (f)lex, vi(m), hexdump, UNIX tools (grep/awk/sed/...), SVN/CVS/MKS/Clearcase, doxygen, javadoc, ...

**Libraries** C++ STL, MFC, OpenGL, X-Windows, Java Sun container classes, Java Swing, ...

**Communication protocols** TCP/IP, HTTP, (E)SMTP, SNMP, CAN, SPI, ...

**Application languages etc.** XML, XSL, SQL, mySQL, Postscript, Prolog, UML, formal specification languages (Z, predicate calculus), ...

**Other relevant skills/knowledge:**

- Good all-round knowledge in physics, electronics, mathematics (incl. numerical and graph algorithms), AI fields (neural networks, natural language processing, text association), and computer science (hashing, random generation, balanced trees, compiler design); basic knowledge in information theory
- Computer science experience includes: Code generation (C/C++, VBA, XSL, XML/HTML, Postscript); Coding of interpreters; Creation of own software tools (e.g. Java file-scope class detector, calling tree analyzers, BASIC variable definition checker); Computer graphics (ray tracing, splines); Object-oriented design (in any programming language); UNIX system programming (pthreads, pipes, ...)

## EDUCATION:

Technical University (TU) Delft, Netherlands, 1986 - 1991: M.Sc. Electrical Engineering

- Specialization: Semiconductor physics and computer simulation of semiconductor devices
- Traineeship during studies (1990, Philips Nijmegen): Computer modeling of ESD protection transistors.

## PROFESSIONAL CURRICULUM VITAE: Oldest jobs first. [Jump to most recent job](#)

- 1992-1996: First jobs, Netherlands
  - 1992 (Delft Institute of Micro-Electronics and Sub-micron technology, Delft) -- Writing a program for computation of charge distributions in SiGe MOS transistors (C++, Apple Macintosh); plus assistance with literature research.
  - 1993-1994 (Warma Engineering, Ridderkerk bei Rotterdam) -- General maintenance of a program (Basic, DOS) for control/monitoring of heating installations.
  - 1995 (INCORE Automatisering, Amsterdam) -- Design and implementation of a program for generic and configurable data communication (file transfer through TCP/IP) between a warehouse management program and a remote database (C, OS/2).
  - 1996 (TNO-FEL, Den Haag) -- Implementation of a program for computation of the infrared contrast of a navy ship, on the basis of a simple model. Also assistance with making the model mathematically consistent. (FORTRAN-90, DOS).
  - 1996 (CMG, Den Haag) -- Programming work for control of the mobile flood-protection dam ("Stormvloedkering") near Rotterdam (UNIX, C/C++).
- 1996-1997 (CONSUL Risk Management B.V., Delft, with TU Delft, Netherlands)  
**Research and software development for anomaly detection for computer security.**
  - BACKGR** The "anomalies" to be detected were defined as: suspicious, untypical entries in a logs of the behaviour of computer network users. The purpose of this anomaly detection was intrusion detection through detection of changed user behaviour.
  - LONG** R&D on detection of "anomalies" in logs of the behaviour of computer network users. Research into data analysis methods, such as statistical methods, that can be used in anomaly detection; followed by design, implementation, documentation, and testing of a prototype program that executes anomaly detection in computer logs. (ANSI C, SQL, Sybase, UNIX, Windows NT, MVS.)
- 1998-2000 (Freelance work, Netherlands)  
**Analyzing and documenting the logical functionality of existing programs, to open up the software to the customer for re-implementation and/or maintenance**
  - LONG** for the following two programs: 1. (for Europe Data Consult, Rotterdam) an old Pascal program for computation of technical specifications of a electrical power conductor; 2. (for RIKS, Ministry of Water Management, The Hague, and EDS, Leidschendam) the DIGIPOL program, a C program for interpolation of sea- and river depth measurement data.
  - Various other assignments:**
    - LONG** (for Rentmeester Informatisering, Alphen a/d Rijn) conversion of Windows '9x printer driver software to Windows NT 4.0; (for EDS Leidschendam) UNIX shell scripting and documentation for a backup system.
- 2001 (ED&T, Philips Research, Eindhoven, Netherlands)  
**Re-implementation of a program for conversion of the model description of a mixed analog/digital IC.**
  - BACKGR** The purpose of this program (called "maketiming") was to convert the model description of a mixed analog/digital IC into an (approximate) fully digital IC, so that the timing behaviour of the mixed analog/digital IC could be computed by simulation software intended for fully digital ICs.
  - LONG** Re-implementation of the program to another internal data structure. This was necessary to enable the software to handle additional digital cell library file formats.  
Additionally: Documentation and debugging of a few auxiliary programs used in translation of digital cell libraries to other file formats. (UNIX, C++)
- 2002-2004 (im-brain GmbH, Dortmund, Germany)  
**Design and development of semantic text association software.**
  - BACKGR** This text association software is a "brain" that can compare texts in a human-like way, by taking into account the (automatically generated) "cloud" of associations around each word.
  - LONG** Main task: Design, implementation and testing of the central text association module used by all im-brain applications.  
Additionally: Research into extension of the association machine to image data, and delivery of prototype systems to compare logo images and to detect blocks of text in scanned pages. Delivery of general-purpose hierarchical clustering software, with graphical visualization of the clustering result. Improvement of the structure of one of the (server-based) end-user programs. In all im-brain software, organization of the code into libraries, and help with making the code more error-free. Writing of software design and algorithm documentation. (Linux, C++)
- 2005 (Docuserve, Hamburg, Germany -- for customer Basler Vision Technologies, Ahrensburg, Germany)  
**Writing a programmer's guide (API overview) for the software library of a "smart camera".**
  - BACKGR** The camera incorporated a general-purpose processor, on which user programs could be run (typically for image processing). Delivered with the camera is a software library, which allows these user programs to control the camera's image capturing and image transfer hardware.

**LONG** Main task: Writing an API overview text (programmer's guide) for this software library. The purpose of this text was to give the application programmer an overview picture over the library's API, which the automatically (doxygen) generated "API Reference" could not give. This task included giving assistance in getting clear (from the programmers and from the code) which methods/classes from the library software actually belonged to the exported API.

Additionally: Help with structuring the "API Reference" generated automatically from comments in the source code; Writing of explanatory/tutorial texts to go with the set of sample programs delivered with the camera. (The software library was a C++ library, using templates and exceptions. The processor on the camera was running the Linux operating system.)

- 2006 (JULIE Lab, University of Jena, Germany)

**Java software development for a natural language processing research group.**

**LONG** Design and implementation of an infrastructure system used for data processing and data storage needed for "supervised learning" of language processing software modules, i.e. adaptation of their parameters to training samples. This system included a central data repository, a server for access to this repository, and client programs (for management of the repository, and for running an external GUI program for entering the expected outputs to selected training samples). (Java, TCP/IP, Linux.)

- Sep 2006 - Dec 2007 (Manu-Dextra GmbH, Nürnberg, Germany -- for customer Automotive Distance Control Systems GmbH, Lindau/Bodensee, Germany)

**Architecture/design documentation, and development, for the microcontroller software in a radar sensor device.**

**BACKGR** The software on the two microcontrollers (Renesas M32192) in this radar sensor device computed from the raw radar data the position and speed of the (nearest) objects in the region swept by the moving radar beam. The software (ANSI C) was strictly modularized into separate software components, much like C++ classes. (These modules largely conformed to the automotive AUTOSAR standard.)

**LONG** My tasks during this assignment consisted to 75% of software documentation, and to 25% of software development:

**Documentation:** Software architecture documentation (description of the application software in the device as a whole, treating the software components as black boxes), both for the radar sensor device, and also (as a separate document) for the general architecture of the suite of basis software components used on all devices manufactured by the company. Design documentation for the boot-loader software on both microcontrollers in the radar sensor device. Design documentation for one of the application software components in the radar sensor device, namely the "ACTL" component (Application ConTroL), responsible for controlling (coordinating) the cyclic data processing in the device.

**Software development:** Extension of the "ACTL" software component, namely making the state machine of the central "operation mode" of the device configurable at compile time via Excel tables. These human-readable Excel tables contain VBA code that generates C source files containing the definition of static C arrays and structs; this static data constitutes the "program" that controls the operation of the state machine during run time. Additionally, writing of Perl scripts for code generation for the software component operating the CAN bus, and customer adaptation of that same CAN component. For the software written/modified by myself, I delivered extensive documentation as well. (C, CAN, OSEK/VDX, Vector/Lauterbach tools, Perl, VBA)

- Jan-Jun 2008

Gap. This period I intentionally I took off work, to experiment at home with 8-bit AVR microcontrollers, in order to deepen my understanding of the hardware aspects and low-level (assembler) programming of microcontrollers.

- Sep 2008 - Jun 2010 (DAKO GmbH, Jena, Germany)

Two tasks:

**Design and implementation of a new 3D-kernel for 3D CAD programs (Main task).**

**BACKGR** The company DAKO owns its own 3D CAD (Computer Aided Design) programs for creation and viewing of catalogs of mechanical components. As in many 3D CAD programs, the CAD user here creates complex components step-by-step, by means of intersection, subtraction, and union operations on simple 3D solids ("Constructive Solid Geometry", CSG).

**LONG** My task was to extend the functionality of these CAD programs so that every solid (3D component) is available not only as a "recipe" of successive CSG operations (which the software could already do), but also as a so-called "boundary representation", namely as the set of connected surface elements (facets) and edges that the surface of the solid consists of.

To accomplish this, I designed and implemented a completely new 3D kernel (software library, software component). The most important task of this 3D kernel is to execute (compute) a CSG operation, where both the two input solids as well as the output solid are all represented by their surface elements and edges; and where the output solid can be fed as input into further CSG operations. Only four surface element types were needed: plane, cylinder, cone, and torus. The 3D kernel software was strictly isolated from OpenGL and from all visual rendering.

Sub-tasks included: Designing datastructures for the surface segments and edges; Computation of the intersection curves between two "arbitrary" surfaces (analytic in simple cases, numeric for complex surface segments); Writing a tessellator usable for curved surfaces; Adding a regression tester (indispensable because of the complexity of the software, and the large quantity of mathematical special cases). (C++, OpenGL, MS Visual C++, Windows PC)

**Programming of a parser and interpreter for a new script language (2 months).**

**BACKGR** In order to allow the manufacturer of the mechanical components to program the configuration of his catalogs himself, DAKO decided to develop a new, BASIC-like scripting language.

**LONG** Assistance with fixing an exact grammar; Documentation of the grammar; Hand-coding of a "top down" parser (predictive parser) and of an interpreter. The script language included functions and local variables. Parser and interpreter implemented as strictly separate components (and each isolated from the client code by C++ "interfaces"). (C++, MS Visual C++, Windows PC)

- Oct 2010 - May 2011 (e.sigma Systems GmbH, München, Germany)

**Documentation, source code analysis, and maintenance for a military simulation installation.**

**BACKGR** The company e.sigma delivers and services military training and simulation installations. I worked on location in their biggest simulation installation, delivered to the German army. The purpose of this installation is to test guidance systems used in missiles, i.e. to test whether the sensor and image/data processing electronics in the missile is able to detect and track targets successfully. The installation consists of a spherical projection surface (radius 20 m) that can reflect both visual and infrared wavelengths; the device under test is placed in the center of the sphere, mounted on a 3-axis turn table. Virtual scenarios (landscapes with moving targets) are presented to the device under test, from a battery of 27 visual projectors and from three IR laser guns mounted on the inner axis of a 2-axis turn table.

The installation was controlled by around 50 separate PCs, each with its own special task. These PCs were networked via 3 separate Ethernet networks (TCP/IP). Additionally, a Reflective Memory network provided "real time" communication between most of these PCs (apart from the image generators); the PCs thus interconnected all used Real Time Linux (= RedHat Linux with real time patch).

**LONG** The following **Documentation** was delivered by me: Documentation about the central program coordinating the whole installation (logic, data flows, program states, run of a simulation). A big diagram showing all PCs and all other units in the installation and how they are interconnected (purpose: providing insight into the installation as a whole). Documentation about a few auxiliary programs (for configuration of the Reflective Memory/DAQ communication; for distribution of the image data into the installation), and about how to boot up and power down the installation.

**Source code analysis** of the source tree for the programs responsible for the simulation and projection of the visual and IR targets. These sources had been written for e.sigma by a third-party company, which had left behind a huge source tree (2182 subdirectories, 621 makefiles, 12048 C/C++ source files), scarcely documented and commented, about whose internal operation e.sigma almost had no knowledge at all. In this source tree, I identified the obsolete 60% and the compilable 40%, and in the latter the sources for both programs and for the libraries used by them. Finally I made sure that the executables built from these sources were identical to the executables installed by ECOM (same symbols).

**Maintenance:** Porting to a new PC with a new Linux version of existing bash scripts for control of the UPSes and for booting up/shutting down the installation (about 10 scripts, using lockfiles, samba, snmpget/set); updated and extended the existing minimal installation instruction document. Answering questions of the German army about the installation and about the software. Receiving their wishes for extensions/changes, and forwarding these to e.sigma. (bash, C/C++, GNU tools, TCP/IP, SNMP, networked RTLinux and Windows PCs.)

Legend:

**BACKGR** Technical background information (context) about the assignment

**LONG** Full description of the work delivered.