

Name	Menno RUBINGH
Beruf	F&E Software-Entwerfer, F&E Software-Dokumentalist
Kontaktdaten	Postanschrift Jansonstraße 18, 07745 Jena Email hgnibur@freenet.de Telefon (Festnetz) +49 . 3641 . 217678 Telefon (Handy) +49 . 174 . 8215421
Geburtsdatum	7 Dez 1966
Staatsangehörigkeit	niederländisch
Sprachen	English -- fließend Deutsch -- gut Niederländisch -- Muttersprache Französisch, Italienisch -- verstehen (Japanisch, Chinesisch -- einige wenige Basiskenntnisse)
Persönlichkeits-Typ	Big Five : O++ C- E-- A- N= MBTI : INTP
Internet	Profil deutsch (HTML) http://www.rubinghscience.org/cv/profil_d.html Profil deutsch (PDF) http://www.rubinghscience.org/cv/profil_d.pdf Profil englisch (HTML) http://www.rubinghscience.org/cv/profile_en.html Profil englisch (PDF) http://www.rubinghscience.org/cv/profile_en.pdf Diplom, Zeugnisse http://www.rubinghscience.org/cv/bew_unterl.html

ZUSAMMENFASSUNG:

Ich bin nützlich einsetzbar überall wo Neues verstanden werden muss; und dann ausgebaut werden muss zu einem funktionierenden Prototyp, und/oder klar beschrieben werden muss. Meine Stärke ist der kreative Weg von vage formuliertem Problem zu funktionierender technischer Lösung, sowie auch das übersichtlich zugänglich machen von komplexen Algorithmen und/oder Quellcode.

ANGEBOTENE DIENSTE:

- **Software-Entwurf** (Software-Struktur und Algorithmik)
 - Ausforschen neuer Möglichkeiten, und Ausarbeiten davon bis zu funktionierender Prototyp-Software
 - Entwurf/Anpassung/Engineering der konkreten Algorithmen, die benötigt sind, die vom Auftraggeber erwünschte Funktionalität zu realisieren (inkl. numerische und KI-Algorithmen)
 - Verbesserung der Struktur von existierendem Code, zur Verbesserung der Änderbarkeit/Erweiterbarkeit
 - Entwurf von Software-Komponenten die Plattform-unabhängig, maximal fehlerfrei, und übersichtlich strukturiert sind
 - Erstellung von Software-tools.
- **Software-Dokumentation** (Design/Algorithmik-Dokumentation)
Erstellung von Texten die
 - Einsicht übermitteln in die Logik, Mechanismen, und Algorithmen benutzt in einer Software
 - eine Übersicht geben über das interne Design einer Software
 - die Verbindung machen zwischen der unterliegenden Logik und dem konkreten Quellcode.

STICHWÖRTER TECHNISCHE ERFAHRUNG:

Programmiersprachen 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 mit 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, ...
Kommunikations-Protokolle TCP/IP, HTTP, (E)SMTP, SNMP, CAN, SPI, ...
Anwendungssprachen usw. XML, XSL, SQL, mySQL, Postscript, Prolog, UML, formale Spezifikationssprachen (Z, predicate calculus), ...

Weitere relevante Fähigkeiten/Kenntnisse

- Umfangreiches Wissen in Physik, Elektronik, Mathematik (inkl. numerische Mathematik und Graph-Algorithmen), KI-Bereichen (neuronale Netze, natural language processing, Text-Assoziation), und Informatik (hashing, random generation, balanced trees, compiler design); Basiskenntnisse Informations-Theorie
- Informatik-Erfahrung umfasst u.a.: Code-Generierung (u.a. von C/C++, VBA, XSL, XML/HTML, Postscript); Kodierung von Interpretern; Erstellung eigener Software-tools (z.B. Java file-scope class detector, calling tree analyzers, BASIC variable definition checker); Computer-Graphik (ray tracing, splines, NURBS); Objekt-orientierte Entwicklung (in beliebigen Programmiersprachen); UNIX System-Programmierung (pthreads, pipes, ...)

AUSBILDUNG:

Technische Universität (TU) Delft, Niederlande, 1986 - 1991: Diplom Elektrotechnik

- Spezialisierung: Halbleiterphysik und Computer-Simulation von Halbleiter-Elementen
- Praktikum während des Studiums (1990, Philips Nijmegen, Niederlande): Computer-Modellierung von ESD-Schutz-Transistoren.

BERUFLICHES CURRICULUM VITAE: Älteste Jobs zuerst. [Gehe zum letzten Job](#)

- 1992-1996: Erste Jobs, Niederlande
 - 1992 (Delft Institute of Micro-Electronics and Sub-micron technology, Delft) -- Schreiben eines Computerprogramms für Berechnung von Ladungs-Verteilungen in SiGe MOS Transistoren (C++, Apple Macintosh); und Assistenz bei Literatur-Recherche.
 - 1993-1994 (Warma Engineering, Ridderkerk bei Rotterdam) -- Allgemeine Wartung an einem Programm für Steuerung/Überwachung von Heizungs-Installationen. (Basic, DOS.)
 - 1995 (INCORE Automatisering, Amsterdam) -- Entwurf und Implementierung eines Programms for generische und konfigurierbare Daten-Kommunikation (file transfer über TCP/IP) zwischen einem Lagerhaus-Management-Programm und einem remote database. (C, OS/2.)
 - 1996 (TNO-FEL, Den Haag) -- Implementierung eines Programms for Berechnung des Infrarot-Kontrasts eines Marine-Schiffes, basiert auf ein einfaches analytisches Modell. Dabei auch mitgeholfen das Modell mathematisch konsistent zu machen. (FORTRAN-90, DOS.)
 - 1996 (CMG, Den Haag) -- Programmierarbeit an der Steuerung des Sturmflut-Schutz-Damms ("Stormvloedkering") bei Rotterdam. (UNIX, C/C++.)
- 1996-1997 (CONSUL Risk Management B.V., Delft, Niederlande; in Zusammenarbeit mit der TU Delft, Niederlande)
F&E an anomaly detection für computer security.
 - BACKGR** Die zu detektierenden "anomalies" wurden definiert als: verdächtige, nicht-übliche Einträge in logs (Aufzeichnungen) des Verhaltens von Benutzern von Computer-Netzwerken. Das Ziel dieser anomaly detection war das Erkennen (möglicher) Intrusion in das Netzwerk aus Änderung des Verhaltens der Benutzer.
 - LONG** Forschung nach für anomaly detection benutzbare Daten-Analyse-Methoden (u.a. statistische Methoden); danach Entwurf, Entwicklung, Dokumentation, und Testen eines Prototyp-Programms für anomaly detection in computer logs. (ANSI C, SQL, Sybase, UNIX, Windows NT, MVS.)
- 1998-2000 (Freiberufliche Arbeit, Niederlande)
Analyse und Dokumentation der logischen Funktionalität existierender Programme, um den Quellcode für den Kunden zu öffnen für Änderung und Weiterentwicklung
 - LONG** für die folgenden Programme: 1. (für Europe Data Consult, Rotterdam) ein altes Pascal-Programm für Berechnung der technischen Spezifikationen eines elektrischen Leistungs-Stromleiters; 2. (für RIKS, Ministerium für Wasserbeherrschung, Den Haag) das DIGIPOL Program, ein C Programm für Interpolation von Meeres- und Fluss-Tiefenmessdaten.**Sonstige Aufträge:**
 - LONG** (für Rentmeester Informatisering, Alphen a/d Rijn) manuelle Konversion von Windows '9x printer driver Software nach Windows NT 4.0; und (für EDS Leidschendam) UNIX shell scripting und Dokumentation für ein Backup-System.
- 2001 (ED&T, Philips Research, Eindhoven, Niederlande)
Neu-Implementierung eines Programs für Konversion der logischen Modellbeschreibung von analogen/digitalen ICs.
 - BACKGR** Das Programm (genannt "maketiming") diente dazu, die Modell-Beschreibung eines gemischt analog/digitalen ICs zu konvertieren nach ein angenähertes völlig digitales Modell, so dass Simulations-Software für digitale ICs eingesetzt werden konnte für die Berechnung des Zeit-Verhaltes eines gemischt analog/digitalen ICs.
 - LONG** Neu-Implementierung des Programms zu einer anderen internen Datenstruktur. Dies war notwendig für die Erweiterung des Programms zu Verarbeitung von zusätzlichen File-Formaten für digital cell libraries. Zusätzlich: Dokumentation und Fehlersuche für einige Hilfs-Programme benutzt bei der Konversion von digital cell libraries nach anderen Datei-Formaten. (UNIX, C++)
- 2002-2004 (im-brain GmbH, Dortmund)
Entwurf und Entwicklung von Text-Assoziations-Software.
 - BACKGR** Diese Text-Assoziations-Software ist ein "Gehirn" das Texte auf "Mensch-ähnliche" Weise mit einander vergleichen kann, dadurch dass es (automatisch generierte) "Wolken" von Assoziationen rund jedes Wort berücksichtigt.
 - LONG** Hauptaufgabe: Entwurf, Implementierung und Testen des zentralen Assoziations-Moduls benutzt von allen im-brain Anwendungs-Programmen.
Weitere Aufgaben: Forschung nach Erweiterung der Assoziations-Maschine zu Bild-Daten, und Auslieferung eines Prototyp-Systems dass Logo-Bilder vergleicht und Text in eingescannten Bildern erkennt. Herstellung von general-purpose Hierarchische-clustering-Software, mit graphischer Visualisierung des Ergebnisses der Clustering. Verbesserung der internen Struktur eines der (server-basierten) end-user Programmen. Zerlegung des im-brain Quellcode in Bibliotheken, und Hilfe geleistet dabei, den Code mehr fehlerfrei zu machen. Schreiben von Software-Design- und Algorithmen-Dokumentation. (Linux, C++)

- 2005 (freiberuflich für Docuserve, Hamburg -- für Kunde Basler Vision Technologies, Ahrensburg)
Schreiben eines "Programmer's Guide" für die API der Software-Bibliothek einer "smart camera".
BACKGR Die Kamera war ausgestattet mit einem general-purpose Prozessor, worauf Anwender-Programme (u.a. zur Bildverarbeitung) ausgeführt werden konnten. Über eine bei der Kamera mitgelieferten Software-Bibliothek können diese Anwender-Programme die Kamera steuern (z.B. Starten von Aufnahmen, Konfiguration von Beleuchtungs-Parametern).
LONG Hauptaufgabe: Schreiben einer Übersicht ("Programmer's Guide") über die API dieser Software-Bibliothek. Zweck dieses Dokumentes war, dem Anwendungs-Programmierer (beim Kunden von Basler) die Übersicht und Einsicht in den Zusammenhängen zu geben die die automatisch (mit doxygen) aus dem Quellcode generierte Dokumentation nicht übermitteln konnte. Zur Arbeit gehörte auch, mitzuhelfen zu klären (mit den Programmierern und aus dem Quellcode) welcher Teil der Methoden/Klassen im Library eigentlich zur exportierten API gehörte.
Zusätzlich: Assistenz bei der Strukturierung des automatisch aus dem Quellcode generierten Teils der Dokumentation; Schreiben von erklärenden Beschreibungen bei den mit der Software-Library mitgelieferten Sammlungen code samples (Beispiel-Anwenderprogrammen). (Die Software-library war eine C++ library, mit Benutzung von templates und exceptions. Auf dem Prozessor in der Kamera lief das Linux Betriebssystem.)
- Jan-Mai 2006 (JULIE Lab, Universität Jena)
Java Software-Entwicklung für eine Sprachverarbeitungs-Forschungsgruppe.
LONG Entwurf und Implementierung eines Infrastruktur-Systems für die Datenverarbeitung und Datenspeicherung benötigt für "überwachtes Lernen" von Sprachverarbeitungs-Software-Modulen, d.h. Adaptation deren Parameter an Trainings-Mengen. Das System umfasste eine zentrale Daten-Repository, einen Server für Zugriff auf diese Repository, und Client-Programme (für Verwaltung der Repository, und für Aufruf eines externen GUI-Programmes womit die Soll-Werte auf selektierte Trainings-Mengen eingegeben wurden). (Java, TCP/IP, Linux.)
- Sep 2006 - Dez 2007 (Manu-Dextra GmbH, Nürnberg -- für Kunde Automotive Distance Control Systems GmbH, Lindau/Bodensee, einen Teil der Firma Continental)
Dokumentation und Entwicklung für die Microcontroller-Software in einem Radar-Sensor-Gerät.
BACKGR Die Software auf den zwei Microcontrollern (Renesas M32192) in diesem Radar-Sensor-Gerät berechnete aus den rohen Radar-Daten die Position und Geschwindigkeit der (nähesten) im vom Radarbündel durchzogenen Raumabschnitt sich befindenden Objekte. Die Software (ANSI C) war strikt modularisiert in separate Software-Komponenten, einigermaßen ähnlich zu C++ Klassen. (Diese Komponenten entsprachen weitgehend dem automotive AUTOSAR Standard.)
LONG Meine Aufgaben bestanden zu etwa 75% aus Dokumentation, und zu etwa 25% aus Software-Entwicklung:
Dokumentation: Software-Architektur-Dokumentation (Beschreibung des Entwurfs der gesamten Applikations-Software im Gerät, mit den Software-Komponenten als 'black boxes'), sowohl für das Radar-Gerät als auch separat für die Architektur der auf allen Geräten benutzten Basis-Software-Komponenten. Design-Dokumentation für die Boot-loader-Software auf beiden Microcontrollern im Radar-Sensor-Gerät. Design-Dokumentation für eine der Software-Komponenten in der Applikations-Software des Radar-Sensor-Geräts, nämlich die "ACTL"-komponente (Application ConTroL), zuständig für die Überwachung und zeitliche Steuerung der zyklischen Datenverarbeitung im Radar-Gerät.
Software-Entwicklung: Erweiterung der "ACTL"-Software-Komponente, nämlich das zur compile-time über Excel-Tabellen konfigurierbar machen der Zustandsmaschine für die zentrale "operation mode" des Radar-Geräts. Diese (für Menschen lesbare) Excel-Tabellen enthalten VBA Code, der C Source-Files generiert mit darin die Definition von statischen C arrays und structs; diese statische Daten sind das "Programm" das während Laufzeit das Verhalten der state machine bestimmt. Zusätzlich Erstellung von Perl-scripts für Code-Generierung für die Software-Komponente für CAN-Kommunikation, und Kunden-Anpassung in der selben CAN-Komponente. Auch für diese von mir geänderte/erstellte Code und Scripts habe ich ausführliche Dokumentation abgeliefert. (C, CAN, OSEK/VDX, Vector/Lauterbach tools, Perl, VBA)
- Jan-Jun 2008
Lücke. Diese Zeit habe absichtlich frei genommen und zu hause experimentiert mit 8-bit AVR Microcontrollern, um meine Einsicht zu vertiefen in die hardware und die assembler-Programmierung von Microcontrollern.
- Sep 2008 - Jun 2010 (DAKO GmbH, Jena)
Zwei Aufgaben:
Entwurf und Implementierung eines neuen 3D-Kerns für 3D CAD-Programme (Hauptaufgabe).
BACKGR Die Firma DAKO hat ihre eigene 3D CAD-Programme (Computer Aided Design) für das Erstellen und Anzeigen von Katalogen von mechanischen Bauteilen. Wie in vielen 3D CAD Programmen wird auch hier so vorgegangen dass der CAD-Zeichner ein komplexes Bauteil schrittweise zusammenstellt durch das Schneiden, Subtrahieren, und Zusammenfügen von einfachen 3D-Körpern ("Constructive Solid Geometry", CSG).
LONG Meine Aufgabe war, die Funktionalität dieser CAD-Programme zu erweitern so dass jeder 3D-Körper (Bauteil) verfügbar ist nicht nur in einer Repräsentation als "Rezept" von sukzessiven CSG-Operationen (was die Software vorher schon konnte), sondern auch als sogenannte "boundary representation", nämlich als Menge der verbundenen Oberflächen-Segmente (Fazetten) und Kanten woraus die Oberfläche des Körpers besteht.
Dazu habe ich einen völlig neuen 3D-Kern entworfen und implementiert. Die wichtigste Funktion dieses 3D-Kerns ist, eine CSG-Operation ausführen (zu berechnen), wobei sowohl die zwei input-Körper als auch der output-Körper repräsentiert werden durch ihre Oberflächen-Segmente und Kanten; und wobei der output-Körper als input für weitere CSG-Operationen verwendet werden kann. Benötigt waren hierbei nur 4 Typen von Oberflächen-Segmenten: Ebene, Zylinder, Kegel, und Torus. Die 3D-Kern Software wurde strikt isoliert von OpenGL und von jeder visuellen Darstellung.
Teilaufgaben: Neue Datenstrukturen entwerfen für die Oberflächen-Segmente und Kanten; Die

Schnittkurven berechnen zwischen zwei "beliebigen" Oberflächen (analytisch für einfache Fälle, numerisch für komplexeren Oberflächen); Einen Tessellator schreiben der einsetzbar ist für gebogene Oberflächen; Einen Regressions-Tester hinzufügen (war unerlässlich wegen der Komplexität der Software, und wegen der grossen Menge der mathematischen Spezialfälle). (C++, OpenGL, MS Visual C++, Windows PC)

Programmierung Parser und Interpreter für eine neue Script-Sprache (2 Monate).

BACKGR Mit dem Ziel, es dem Bauteile-Hersteller zu ermöglichen die Konfiguration seine Kataloge selbst zu programmieren, entschied DAKO zur Entwicklung einer neuen, BASIC-ähnlichen Script-Sprache.

LONG Hilfeleistung dabei, die exakte Syntax der Sprache festzulegen (zu entscheiden); genaue Dokumentation der Syntax; Hand-Kodierung eines Parsers ("top down" parser, predictive parser) und eines Interpreters für diese Sprache. Die Script-Sprache umfasste Funktionen und lokale Variablen. Parser und Interpreter implementiert als strikt getrennte Teile (und beide durch C++ "interfaces" vom client code abgeschottet). (C++, MS Visual C++, Windows PC)

- Okt 2010 - Mai 2011 (e.sigma Systems GmbH, München)

Dokumentation, Source-Code-Analyse, und Wartung für eine militärische Simulationsanlage.

BACKGR Die Firma e.sigma liefert und wartet militärische Trainings- und Simulations-Anlagen. In ihrer größten Simulationsanlage, geliefert an die deutsche Bundeswehr, habe ich für e.sigma vor Ort gearbeitet. Die Anlage hat den Zweck, Suchköpfe aus Flugkörpern zu testen, d.h. zu überprüfen ob die Sensorik und die Bild/Datenverarbeitungselektronik im Suchkopf erfolgreich Ziele detektiert und verfolgt. Die Anlage besteht aus einer kugelförmigen Projektionsfläche (Radius 20m) die sowohl visuellen als auch IR-Wellenlängen reflektieren kann; Der Prüfling (Suchkopf, device under test) wird im Mittelpunkt der Kugel aufgestellt, auf einem 3-Achser Drehtisch. Auf der Kugeloberfläche werden dem Prüfling virtuelle Szenarien (Gelände und darin bewegende Ziele) projiziert, aus einer Batterie von 27 visuellen Projektoren, und aus drei IR-Laserkanonen (für verschiedenen IR-Wellenlängen) montiert auf der inneren Achse eines 2-Achser Drehtisches.

Für die Steuerung benutzte die Anlage ca. 50 separaten PCs, jeder mit seiner eigenen speziellen Aufgabe. Diese PCs waren vernetzt über 3 separaten Ethernet-Netzwerke (TCP/IP). Für "Echtzeit"-Kommunikation waren zudem die meisten dieser PCs (ausser den Bildgenerator-PCs) miteinander verbunden über ein Reflective-Memory-Netzwerk; die so verbundenen PCs benutzten alle Real-Time-Linux (= RedHat Linux mit real time patch).

LONG Folgende **Dokumentation** habe ich erzeugt: Dokumentation über die Grundlagen des Programms dass die ganze Anlage zentral steuert (Logik, Datenflüsse, Zustände, Ablauf einer Simulation). Einen grossen Plan (Schema) worauf alle PCs und sonstige Hardware der Anlage dargestellt ist, zusammen mit allen Netzwerken (Ziel: Grundlage für Besprechungen über die Anlage, und für Einarbeitung weiterer Mitarbeiter). Dokumentation über einige Hilfsprogramme (für Konfiguration der Reflective-Memory/DAQ-Kommunikation; für Verteilen der Bilddaten-Dateien in die Anlage), und über das Hoch- und Runterfahren der Anlage.

Source-Code-Analyse: Analyse der source tree für die Programme verantwortlich für die Simulation und Darstellung der visuellen und IR-Ziele. Diese Sourcen waren in Auftrag von e.sigma geschrieben worden von einer Drittfirma; die eine riesige source tree hinterlassen hatte (2182 Unterverzeichnisse, 621 Makefiles, 12048 C/C++ source files), kaum dokumentiert und spärlich kommentiert, über die e.sigma kaum eigenes Wissen gehabt hat. In der source tree die veraltete 60% und aktuelle 40% identifiziert, und in letzterer die sourcen für die zwei Programme und die von ihnen benutzte libraries; schliesslich überprüft dass die daraus neu gebildeten executables gleich waren zu den von der drittfirma hinterlassenen executables (gleiche symbols).

Wartung: Existierende bash scripts für Steuerung der USVen und An/Abschalten der Anlage portieren auf einen neuen PC mit einer neuen Linux-Version (ca. 10 scripts, mit Benutzung von lockfiles, samba, snmpget/set); die existierende minimale Installationsanleitung erneuert und erweitert. Fragen der Bundeswehr über die Anlage und die Software beantworten. Ihre Erweiterungs/Änderungswünsche entgegennehmen und an e.sigma weiterleiten. (bash, C/C++, GNU Entwicklungstools, TCP/IP, SNMP, vernetzte RTLinux und Windows PCs.)

Legende:

BACKGR Einige technische Hintergrund-Infos zu der Aufgabe (zur Verständlichkeit)

LONG Ausführliche Beschreibung der gelieferten Arbeit.