Curriculum Vitae

**Arbeidsverleden**

**2007 - Heden Senior Softwaredeveloper - Applicatiebeheerder**

**Bedrijf: XXXX**

**Verantwoordelijk voor:**

* Data analyse
* Ontwikkeling en beheer van zowel desktop als webapplicaties
* Architectuur van de applicaties
* Implementeren statistische en wiskundige modellen in de software

**Gewerkt met:**

* VB.NET
* C#
* ASP.NET
* SQL-server
* Oracle

**2005 – 2007**

**Softwaredeveloper**

**Bedrijf: XXX**

**Verantwoordelijk voor:**

* Implementeren van een omvangrijke multi-tier wep appplicatie in Java (J2EE)

**Gewerkt met:**

* Java 2 (o.a. J.S.P.'s, Servlets, het Cocoon-framework, E.J.B.'s)
* Oracle
* SQL-server

**September 2003 -2005**

**Softwaredeveloper / Consultant**

**Bedrijf: XXX**

**Verantwoordelijk voor:**

* Het bouwen van op XML gebaseerde applicaties voor een breed scala klanten, voorbeelden hiervan zijn:
  + een webbased Documenten database waaraan een XML-database ten grondslag ligt (X-hive)
  + een op Oracle gebaseerd Content Management Systeem
  + data-integratie applicaties (data die vanuit zeer diverse bronnen wordt aangeleverd en naar een gestandaardiseerd xml-formaat wordt getransformeerd).

**Gewerkt met:**

* XML en XSLT (waarbij ook veevuldig van XML-schema’s en dtd’s gebruik werd gemaakt)
* J2EE (m.n. J.S.P.'s en, Servlets)
* COM (Visual Basic enDelphi),
* SOAP
* Active Server Pages
* Java- en VBScript
* Oracle
* Klanten van Salience waarvoor ik opdrachten heb uitgevoerd zijn onder andere: Shell, Philips Medical Systems, de Technische Universiteit Delft, De Nederlandsche Bank, AKZO en de R.O.B. (Rijksdienst voor het Oudheidkundig Bodemonderzoek)

**Maart 2003 – September 2003**

**Webdeveloper**

**Bedrijf: XXX**

**Verantwoordelijk voor:**

* Het ontwikkelen van een webapplicatie ten behoeve van de WOZ-taxatie

**Gewerkt met:**

* Active Server Pages
* JavaScript (client-side) XML en XSLT (waarbij ook veevuldig van XML-schema’s en dtd’s gebruik werd gemaakt)
* de Back-office (de “Business logica”) werdt gevormd door Windows Script Componenten (dit zijn objecten die in een scripttaal zijn geschreven maar functioneren als volwaardige COM/ActiveX-componenten) die communiceren met verschillende databases. Zie appendix

**1996 - 2003**

**Senior Software Engineer**

**Bedrijf: XXX**

**Verantwoordelijk voor:**

* onderzoek en ontwikkeling t.b.v. productvernieuwing
* het ontwikkelen van zowel desktop als webapplicaties
* project management

**Gewerkt met:**

* Visual Studio 6 (vooral Visual Basic en Visual InterDev)
* COM-based frameworks
* Active Server Pages
* JavaScript
* Java 2 Enterprise Edition (incl. Enterprise Beans, Servlets, JSP’s, XML en XSL).

**1995 - 1996**

**Bedrijf: XXX**

**Verantwoordelijk voor:**

* Het ontwikkelen van applicaties voor m.n.de Accountancy-branche en de Gezondheidszorg in de ontwikkelomgeving "Progress"

**Gewerkt met:**

* Progress 6 en 7

**1994 – 1995**

Voor verschillende instanties gewerkt als vrijwilliger: De wetenschapswinkel verbonden aan de Erasmus Universiteit Rotterdam (economisch onderzoek), Culotte B.V. (een producent van biologisch afbreekbare wegwerpluiers, bedrijfseconomisch advies).

**Relevant Ervaring**

VB- en C#.Net 7 jaar

JavaScript (client-side) 7 jaar

Active Server Pages 6 jaar

Java 2 (o.a. J.S.P. and Servlets en Enterprise Java Beans) 4 jaar

Borland Enterprise Server 5 1,5 jaar

Visual Basic 5 jaar

Visual InterDev 5 jaar

Visual J++ 1 jaar

(D)HTML 6 jaar

(D)COM-technology 4 jaar

SQL-Server 10 jaar

Oracle 3 jaar

Progress 7 RDBMS & 4GL 6 maanden

Alle MS Windows versies

MS Office

Enige ervaring met UNIX/LINUX/VAX-VMS

Verscheidene Statistische en mathematische Pakketen (Matlab, SPSS, SAS, TSP)

**Opleiding e.d.**

**1988** – **1994**

Doctoraal Examen aan de  **Erasmus Universiteit Rotterdam**

Kwantitatieve Bedrijfseconomie

**1985 – 1986**

Dienstplicht vervuld als Soldaat Gewondenverzorger bij de 41ste Pantserbrigade in Seedorf, Duitsland

**1977 – 1983**

V.W.O.

**Relevant cursussen:**

Applicatie Ontwikkeling met Progress 6 & 7 4GL (1995)

Applicatie Ontwikkeling voor het Hiscom-ZIS (Ziekenhuis Informatie Systeem (1996)[[1]](#footnote-1).

Webontwikkeling met JavaScript (1997)

De programmeertaal Java (1999)

Het gebruik van XML en XSL bij web- en applicatie-ontwikkeling (1999)

Applicatie Ontwikkeling met Visual Basic (2000)

Applicatie Ontwikkeling met Java 2 (2001)

Applicatie Ontwikkeling met JBuilder4 (2001)

Webontwikkeling in Java 2 (m.n. Servlets en JSP’s)

Installatie, Deployment en Development in J2EE (een cursus met name gericht op het ontwikkelen en deployen van EJB’s, waarbij de Borland Enterprise Server 5.0 als Application Server werd gebruikt)

Webload (een uitgebreid pakket voor het testen van webbased applicaties)

X-hive (een native XML-database, in deze cursus werd geleerd hoe met behulp van Java-applicaties de X-hive database benaderd kan worden)

**Talen**

Nederlands: vloeiend

## Engels: vloeiend

Duits: redelijk

Frans: enigszins

Grieks: spreken redelijk, schrijven enigszins

**Persoonlijke Informatie**

Geboortejaar: 1964

Nationaliteit: Nederlandse

Omgeving: Haaglanden

Rijbewijs: B-E (in het bezit van eigen auto)

**Appendix: korte beschrijving van verschillende projecten.**

Bedrijf XXX maakt software voor (doorgaans grote) verzekeringsmaatschappijen. Ik werk voornamelijk aan een programma genaamd IFM (Integral Financial Modeling) waarmee bv. ziektekosten- en schadeverzekeraars hun toekomstige uitgavenpatronen kunnen inschatten d.m.v. het schatten van een model met de op de tot dan toe bekende data en op basis daarvan (met het programma) weer allerlei soorten analyses kunnen doen (wat zullen de “cash flows” zijn?, hoe staat met de toekomstige solvabiliteit?, hoe goed heeft het geschatte model het voor de afgelopen jaren gedaan? (m.a.w. hoe betrouwbaar zijn de uitkomsten). Deze data zijn veelal betalingen maar kunnnen ook zgn. "incurred" gegevens zijn ("gereserveerde ingeschatte schadebedragen, dit is meestal van toepassing op lang lopende verzekeringen) of gemelde aantallen schade, etc. (er worden in totaal vijf verschillende soorten data onderscheiden)

Het is sterk mathematisch en statistisch georiënteerd wat dus goed aansluit bij de universitaire studie die ik gedaan heb (Kwantitatieve bedrijfseconomie met veel keuzevakken bij Econometrie), de meeste collega's hier zijn ook econometristen. De schattings methode die gebruikt wordt om een model op te lossen ("solven" in IFM-termen) is maximum likelihood estimation over een vrij groot aantal parameters (er zijn tal van verschillende modelinstellingen mogelijk en de data kan op zeer veel verschillende manieren geaggregeerd worden waardoor voor dezelfde data eigenlijk een bijna oneindig aantal modellen te verzinnen is). Om een idee te krijgen er zijn modellen met 4 parameters maar ook met bv. zeventien hetgeen betekent dat er in een 17-dimensionale ruimte geoptimaliseerd moet worden, hetgeen bepaald geen sinecure is en ook erg cpu-intesief is. Daarom heb ik ook een methode ontworpen om het model "asynchroon" te schatten als de verwachting is dat het wel enige tijd kan gaan duren voor de parameters geschat zijn (IFM adviseert dan zelf aan de gebruiker om het asynchroon te doen als hij zelf de checkbox "solve asynchronous" niet heeft aangezet. Het model wordt dan naar een andere webservice gestuurt die het (tijdelijk) opslaat in een database (via Stored Procedures aangezien dit soort modellen normaliter grote hoeveelheden data - in XML-vorm betreffen en dan zijn stored procedures razendsnel). Voor de "server-kant" heb ik een queueing ontworpen waarbij een windows-seervice telkens pollt of er modellen zijn binnnen gekomen om te worden "gesolved" als dit zo is wordt de data en het bijbehorende model (uiteraard weer met een stored procedure opgehaald en door de software van de service gesolved. Zodra een model is gesolved wordt er een email naar de gebruiker gestuurt met de mededeling dat het model kan worden opgehaald (door in IFM "gewoon" op de refresh-knop te drukken, wat hij oook zou doen bij een model dat al een keer gesolved is of bij een model dat in enkele seconden kan worden gesolved. Het grote voordeel is dat de gebruiker na het "asynchroon"wegsturen van een model verder kan gaan met bv. ander werk in IFM (normaliter hebben klanten tientallen tot honderden schadedriehoeken, die weer in een folder-structuur en groepsdriehoeken zijn ondergebracht om het overzichtelijk te houden) of hij kan IFM of zijn PC uit zetten. Als er "gewoon" wordt gesolved IFM zolang het solven bezig is (bij de meeste modellen ca. 3 tot 10 seconden) "unresposive" en gaan de geschatte parameters verloren als IFM tijdens het solven zou worden uitgezet. Dit asynchroon solven is ook heel handig bij het zgn. "batch-solven" dan worden alle driehoeken binnen een bepaalde sectie (maar dit kunnen er bv. 500 zijn) gesolved en indien gewenst zelfs meerdere modellen per driehoek. Dan krijgt de gebruiker een mailtje als de hele batch gesolved is en hoeft hij dus ook niet IFM of zijn systeem aan te laten staan.

Mijn werkzaamheden zijn zowel aan de front-end als aan de back-end. Het is een "klassiek" client-server programma met als mogelijke databases SQL-server, Oracle en MS Access (die laatste eigenlijk alleen voor demo-doeleinden). Het programma maakt echter wel gebruik van webservices (asmx-pagina's) om grote berekeningen op een centrale (heel snelle) server te laten plaatsvinden en de uitkomst ervan terug te laten sturen naar het programma.

De al langer bestaande code is in VB.NET geprogrammmeerd (dit is historisch zo gegroeid, het programma is ooit in een ver verleden als een aantal zeer uitgebreide Excel-macro's begonnen en dan heb je geen andere keus dan VBA (Visual Basic for Applications) te gebruiken en als je dan op een bepaald dus dit levert mij over het algemeen niet veel moeilijkheden op.

We hebben ook een web-applicatie, CMF (Claims Management Filter), zie de website voor informatie) die geschreven is in ASP.NET, waar heb ik ook regelmatig aan meewerk.

De databases die voor IFM nodig zijn onderhoud ik voornamelijk zelf, daarbij moet ik wel zeggen dat de meeste klanten SQL-server gebruiken en dat qua databases mijn meeste kennis daar ligt, maar ik kan dus ook met Oracle uit de voeten (en uiteraard ook met MS Access).

Het is overigens mijn ervaring dat het niet zo heel erg veel uitmaakt welke programmeertaal er wordt gebruikt, met de ervaring die je in de loop der jaren hebt opgebouwd en de huidige ontwikkelomgevingen heb je de syntax van een taal die je wellicht nog niet (goed) kent snel onder de knie. Toen ik bij Posthuma partners kwam had ik nog nooit in .NET geprogrammeerd en het koste me misschien twee a drie dagen om er helemaal vertrouwd mee te raken.

2003- 2005

Bedrijf XXX is een salarisverwerkend bedrijf. Hier werk ik aan een omvangrijke webapplicatie genaamd ESS (Employee Self Service). Dit is een web-appllicatie waarin werknemers zelf hun eigen gegevens kunnen invullen/bijwerken, verlof kunnen aanvragen, overuren kunnen boeken, onkostendeclaraties e.d. kunnen indienen, gegevens over collega’s en andere afdelingen kunnen opzoeken. Het programma werkt met een workflow-server (d.w.z. als er bijvoorbeeld een verlofaanvraag wordt gedaan, komt deze in “de workflow” terecht, de manager van de werknemer die de verlofaanvraag heeft gedaan krijgt automatisch een email en kan dan inloggen op de applicatie (ESS) om de verlofaanvraag te accorderen (of af te keuren), vervolgens wordt de verlofaanvraag eventueel naar een volgende accordeur gestuurd en als alle accordeer-slagen zijn gedaan krijgt de medewerker automatisch een email dat de verlofaanvraag is goedgekeurd (of afgekeurd).

De “Front-end” van dit programma bestaat uit xml-pagina’s (waaruit met gebruik van het Cocoon-framework op run-time html-pagina’s worden gegenereerd) en java-class files waarin de interactie met de gebruiker wordt geregeld. De “middle-tier” wordt gevormd door EJB’s die controles uitvoeren, berichten versturen naar de workflow-server en waarin de connecties met de database (Oracle) worden gemaakt. De “back-end” bestaat dus uit een Oracle Database.

*2003: Ontwikkeling van een webbased*

In dit systeem vormen Active Server Pages de User-Interface en Windows Script Componenten (WSC) de Business-logica. Bij het opstarten van de webapplicatie worden door de global.asa alle WSC’s in de lucht gebracht. Vervolgens zijn deze componenten op afroep beschikbaar in de webapplicatie (a.d.h.v. hun ProgID). Deze WSC’s zijn geschreven in een scripttaal (in ons geval Javascript) met een op xml gebaseerde opzet. Het zijn dus geen gecompileerde componenten maar ze stellen een public interface beschikbaar en functioneren als volwaardige ActiveX/COM-componenten en. Deze WSC’s vormen de verbinding van de user-interface-laag met de database(s) en kunnen – mits goed opgezet - zorgen voor een object georiënteerde manier van programmeren terwijl er toch alleen maar met scripttalen wordt geprogrammeerd hetgeen tevens tot een betere onderhoudbaarheid leidt aangezien er een veel betere scheiding van code en opmaak optreedt.

*2001/ 2002: Ontwikkeling van een Webbased Medisch Zorgwerkstation*

Dit is een omvangrijk web-based werkstation voor verschillende zorgverleners. Dit systeem is gebaseerd op de Architectuur van Java 2 Enterprise Edition (volgens het Model-View-Controller design-pattern.

De user-interface bestaat uit Java Server Pages die echter louter XML en XSL bevatten. Als een gebruiker een actie onderneemt (d.w.z. op een button of hyperlink klikt) wordt er een request naar een java-servlet gedaan. Het servlet (de 'requestcontroller') bepaalt nu welke Enterprise Java Bean moet worden gestart om aan het request te voldoen. De data die door de  Enterprise Java Bean wordt geretourneerd wordt door aan de serverkant omgezet in XML en aan een Java Server Page door gegeven, deze bepaalt nu welk XSL-stylesheet moet worden gebruikt om de data aan de gebruiker te tonen (of om een bevestiging te tonen indien het om een Update, Insert of Delete-actie gaat). Dit systeem bestond ook weer uit een door de gebruiker in te stellen compositie van componenten, dit waren in veel gevallen de webbased variant van de onder het ActiveX-Framework genoemde componenten, die dan echter wel naar het J2EE-platform waren geport (er werd dus niet van ActiveX-componenten gebruik gemaakt).

1. [↑](#footnote-ref-1)