Curriculum Vitae

Name:	Steven Hocking	**Nationality:**	British
Address:	46 Longford Road, Reddish, Stockport, SK5 6UX.	**Availability:**	Immediate
Telephone:	07450432935	**E-mail:**	steve@codemunki.es

Further information

Online CV: http://careers.stackoverflow.com/stevehocking
Github: https://github.com/steve-codemunkies
Web-site: http://www.codemunki.es/

Key Skills

- Extensive experience of **web-based** (web forms and **MVC**) and **desktop development** using **C#** on Microsoft **.NET 1.1, 2** and **4**.
- Extensive development experience with Microsoft **SQL Server 2000, 2005** and **2008**.
- Experience of developing modern modularised web applications with **HTML**, **CSS**, **Javascript** (**Durandal**, **Knockout**, **Moment**, **Cujo.js When**, **Require.js**, and testing with **Jasmine** and **Mocha/Chai**).
- Experience of **Test Driven Development (TDD)** and **Unit Testing** using **NUnit** and the **Microsoft Unit Testing Framework**, including using **RhinoMocks** and **Moq** to provide mocking abilities.
- Experience of setting up and managing **Dependency Injection** using **Castle Windsor**, **Autofac** and **Ninject**.
- Experience of **Git**, **Hg/Mercurial** and **TFS (Team Foundation Server)** source control systems.
- Experience of setting up build environments **MSBuild**.
- Experience of setting up continuous integration using **JetBrains TeamCity**.
- Experience of applying and working with **agile processes** on a number of projects.

Employment history

Havas Health Software (HHS), Manchester – Contract Software Engineer – April 2014 – June 2015 (Extended three times)

- MongoDB
- RabbitMQ
- .NET Framework
    - Moq
    - NancyFx
    - Ninject
    - NUnit
    - TopShelf
- Javascript
    - Durandal
    - Knockout
    - Mocha/Chai
    - Postal.js
    - Require.js

Ukash/Smart Voucher Limited, Bollington, Cheshire – Contract Developer – March 2013 – March 2014 (Extended twice)

- .NET Framework
    - Asp.Net MVC 4
    - AutoFac

    - Moq
    - Nunit
- Microsoft SQL Server
    - Sql Server Integration Services
    - Sql Server Reporting Services

Swinton Insurance, Manchester – November 2011 – February 2013

- Microsoft Message Queue
- Microsoft Sql Server
- .NET Framework
    - Asp.Net MVC 3
    - Castle Windsor
    - nHibernate
    - NUnit
    - Rhino Mocks
    - Windows Communication Foundation (WCF)
    - Windows Forms (WinForm)
- Javascript
    - Jasmine
    - jQuery
    - Knockout

Intechnica LLP, Manchester – October 2010 – November 2011
Overview

- .NET Framework
    - Asp.Net MVC 3
    - Entity Framework 4
    - NUnit
    - Rhino Mocks
- Microsoft SQL Server 2008

2ergo Ltd., Salford Quays – June 2010 – September 2010

- NAnt
- MsBuild
- .NET Framework

PricewaterhouseCoopers LLP (PwC), UK IT Design and Build (D&B), Manchester – July 2004 – June 2010

- Microsoft SQL Server
- Microsoft SQL Server Express
- .NET Framework
    - ADO.NET
    - Asp.Net Web Forms
    - Smart Client Software Factory
    - Windows Forms

Radius Computer Services Ltd, Altrincham – September 1998 – July 2004.

- Microsoft C++
    - MFC
    - ADO
- .NET Framework

Continuing Development

Please see http://careers.stackoverflow.com/stevehocking

Qualifications

Technical Specialist: Web Applications Development with Microsoft .NET Framework 4 (70-515) – May 2011

Microsoft Certified Application Developer – June 2004

- Developing and Implementing Web Applications with Microsoft® Visual C#™ .NET and Microsoft® Visual Studio® .NET – June 2006
- Developing XML Web Services and Server Components with Microsoft Visual C# .NET and the Microsoft .NET Framework – June 2004
- Designing and Implementing Databases with Microsoft® SQL Server™ 2000 Enterprise Edition – April 2004
- Developing and Implementing Windows®-based Applications with Microsoft® Visual C#™ .NET and Microsoft® Visual Studio® .NET – April 2002
- Designing and Implementing Desktop Applications with Microsoft® Visual C++ 6.0 – August 2001

Transcript viewable at https://mcp.microsoft.com/authenticate/validatemcp.aspx (Transcript ID: 663027; Access code: 1976Sh0804).

University of Central Lancashire – BSc Software Engineering (Hons) 2:II – July 1998.

University of Central Lancashire – HND Software Engineering – July 1996.

Additional Information

Havas Health Software (HHS), Manchester

Care4Today is an outcomes-focused solution designed to help deliver quality and customized healthcare when and where our patients need it. Care4Today provides tools and information to patients and caregivers with the goal of improving health outcomes. Novel technologies are applied to healthcare challenges in an effort to increase efficiency and effectiveness through coordination, communication, and connectivity.

Most Valuable Patient Initiative – eyeforpharma – March 2015

Judge's comment: "Guiding a patient from beginning to end of their surgical journey, stepping through all the way to end of recovery? Magnificent. As patients, we often are overwhelmed and unable to absorb required data while preparing for life-changing surgery. This resource appears to be the perfect tool to allow and encourage patients to educate themselves about the intense process at their own pace. Health information such as this will change the future for patients and caregivers worldwide."

Most Valuable HCP or Healthcare Initiative – eyeforpharma – March 2015

Judge's comment: "Excellent project based on deep and meaningful insight, making a real difference for the company, HCPs, and patients"

http://www.eyeforpharma.com/barcelona/awards-index.php

The Care4Today suite of applications are built on a number of technologies:

- All data is stored in a MongoDB document database; there is one database that controls authentication and authorisation for all system users, and determines which Trust's data that user has access to. Each trust then has their own database to store patient's data.
- All applications communicate with the database through a common REST (level 1/2) interface built on NancyFx (.NET, hosted on IIS via OWIN). Certain long running tasks, and generation of documents are handed off to a service process through RabbitMQ (essentially all non-CRUD operations). The web-service is tested through

unit and integration tests written in NUnit, which make use of Moq for mock/stub generation. DI/IoC is achieved using Ninject.
Where the results of long running processes (i.e. doctor/patient letter generation) are required in the web-user interface a message is sent to the client via SignalR.

- The job scheduler is a .NET TopShelf windows service that uses the Quartz Enterprise Scheduler. The job scheduler is responsible for raising timed letters (such as admission letters) and updating patients based date/time (such as admission, expected discharge date, etc.). The service raises messages on to the RabbitMQ to be processed by the service process.
- The service process is a .NET TopShelf windows service that retrieves requests from the RabbitMQ. One of the main jobs of the service is to audit the changes made in the database. The service also generates doctor/patient letters.
- The CPRM HCP web-application is a single page application built using Durandal and Knockout. The application is written in a modern way making use of AMD/Web Modules that are managed through Require.js. Internally the application uses Postal.js to provide bus services (incoming SignalR messages are converted on to the Postal.js bus. Unit tests for CPRM are written using the Given-When-Then pattern using Mocha/Chai.
- The PEM patient web-application provides patients with information pre- and post-surgery, allowing the patient to carry out actions that will aid and speed up recovery. The web-site is built on .NET Umbraco, and uses AJAX to access the common REST service, and then uses Knockout to display data. The web-site is tested through unit tests written in NUnit, which make use of Moq for mock/stub generation. DI/IoC is achieved using Ninject.

During my time at HHS I worked on the Heart Health and Orthopaedic Health streams of Care4today. The majority of my time was focussed on developing new features for the applications, which typically went across all layers of the application. However I did provide significant assistance to other developers (contract and permanent) with creating/updating tests for software (in .NET and Javascript), with a particular area of speciality being Javascript promises. On several occasions I also provided assistance in diagnosing and fixing particularly troublesome defects with the software (the most problematic issues being race conditions that meant that what looked like a single update in the UI was actually multiple updates; these being caused by the evolution of the software).

Ukash/Smart Voucher Limited, Bollington, Cheshire

Ukash is a popular alternative payment brand, designed to help people use cash in the online world - bringing web retailers and cash consumers together in 57 countries on 6 continents. The Ukash secure payment method protects personal identity and financial information when making online transactions - reducing the threat of card fraud and ID theft for consumers, and repudiations and charge-backs for retailers. Ukash voucher codes are available at 465,000 convenient retail locations around the world. Ukash codes are printed on demand at PoS terminals in exchange for cash tendered and can be spent simply online at thousands of participating merchants. Ukash can also be exchanged for a wide range of personal financial services at www.ukash.com, including P2P Payments, International Money Transfer, Prepaid Cards, Gift Cards, Cardless ATM, and more.

Brazilian Real project

The Brazilian Real Project was setup to enable a local partner (Raberil Trading) to sell Ukash vouchers in Brazilian Reals that could only be converted into US Dollar vouchers. Local regulations demand that for each transaction the identity of the person performing the transaction is verified, this is done by checking government issued ID and collecting a value known as a CPF code. This code must be recorded with each purchase.

My task was to create a daily, encrypted extract of all of these transactions, including the amount, the person's name and their CPF code. To do this a Microsoft Sql Server Integration Services (SSIS) package was created. The package extracts the raw data to an unencrypted flat file, and then uses the free gpg program to encrypt the file using Raberil's public key. Finally, because of infrastructure requirements, a custom script component was then used to

e-mail the file through a secured SMTP gateway. This package was then scheduled in SQL Agent to run on a nightly basis.

Cash withdrawal project

The Cash withdrawal project was designed to provide Ukash users with the ability to convert a voucher back into cash. The cash could then be collected through a Bank Machine ATM or PayPoint UK. A key part of this project (to meet UK regulatory requirements) was the introduction of identity checking for users exchanging vouchers for cash.

A large number of small changes were introduced into the User Management System (UMS) used by customer services, a lot of these revolved around the identity verification process (some of which was already being done in a limited fashion), and making customer services administration of this simpler. The UMS Administration application is an MVC 4 application. Part of the work I undertook was to transform the maintainability of the application; this was done by several means:

- Improving the existing use of Dependency Injection - AutoFac was already in use in the application, but registration was idiomatic (each class individually registered, with dependencies explicitly stated).
- New code was developed in a BDD (Behaviour Driven Development) style with GWT (Given - When - Then) testing.
- Where existing code needed to be modified it was refactored to be more SOLID - while the legacy code was typically multi-responsibility it was luckily somewhat de-coupled through the use of interfaces.
- Introduced Jasmine BDD tests for all updated/new javascript.

I also created a Microsoft Sql Server Reporting Services (SSRS) report detailing cash withdrawal transactions and integrate this into UMS. UMS already had a single SSRS report integrated, but this integration was problematic: only one report could be run and the SSRS environment used was hard coded. As part of the changes to integrate the new report into UMS (on the basis that requirements for multiple new SSRS reports were already in the pipeline) I created a new custom configuration section based on System.Configuration.ConfigurationSection. The integration with SSRS was updated so that the report to be executed and the server to do the execution on are loaded from the configuration. This brings the additional advantage that the configuration can be modified through configuration transformations, allowing different reporting environments to be used at different stages of the release process.

Finally I added metrics for the users journey through the cash withdrawal product, this is to enable the business to identify the drop-out rate of those who expressed an interest to actually completing a withdrawal. Because of the complexity of the stored data the best way to retrieve this information turned out to be to use a stored procedure. An SSIS package was created to execute the stored procedure and e-mail the results to interested users within the business. Additionally a new page was added to UMS to enable users with the correct privileges to view the metrics, and run them for any time period of their choosing.

Money transfer (MoneyGram integration) project

The money transfer project was designed to allow Ukash users to transfer money in a controlled manner, via a number of routes. The first phase of the project was to integrate with MoneyGram. I did a number of pieces of work on this project.

I worked to integrate the MoneyGram reversal functionality into the UkashAPI system. This is then used in two different ways:

- Within Ukash API it is occasionally necessary for the system to reverse a MoneyGram transaction because a fault on the Ukash side prevents a transaction being completed, the integration allows this reversal to happen seamlessly within the Ukash transaction, and the user does not need to involve anyone else.
- A user can contact customer services and request that a transfer be reversed.

UkashAPI is a C# web-service, developed in a traditional monolithic style. Working with other contract developers I helped to introduce SOLID concepts into the code, updating some of the common internal services so that they can be easily re-used, rather than being repeatedly redeveloped. This work meant that it was possible to develop the actual code to do the reversal in a BDD style, and introduce some tests around the core services of the web-service.

I also integrated the UkashAPI into the UMS system. Up to this point any functions that the UMS needed to carry out were done directly against databases. However for MoneyGram reversals it was decided that to reduce duplication, and complexity UMS should perform the reversal through the UkashAPI. I therefore had to correctly integrate the UkashAPI service into UMS, and then add in the required functionality. This included pages to allow the customer service user to view details of a user's transactions, view the status of a MoneyGram transfer, and then if necessary action the reversal with the data required by MoneyGram.

Because of the way that the UkashAPI is designed (a single method accepting strings containing XML conforming to *n* of *m* different schemas) a simple "add service reference" did not suffice for the integration. Two new assemblies were added to the project; the first assembly purely provided a set of business entities that could be fed into the proxy assembly. The proxy assembly makes extensive use of generic interfaces and generic base classes so that the combination of business classes determines the type of transaction being performed. The proxy assembly also provides mapping between the return codes supplied from the Ukash API service, and automatic management of the Ukash API sessions.

I also produced a set of SSRS reports providing information on the MoneyGram transactions and their status within the system.

Finally I over-hauled the Store Locator service, to integrate with MoneyGram's Agent Location services. The Store Locator is used by Ukash and external partners to search for nearby locations where Ukash vouchers can be purchased. As part of the MoneyGram integration it was necessary to provide information on locations where transfers can be collected. The first step in the process was to refactor the code of the service. Because the outputs for known inputs was known this was made simpler by introducing a number of integration tests. Once these tests were in place I was able to set about refactoring the code following SOLID principles. The refactored service was then deployed to test environments to ensure that it worked as intended. Next I continued to refactor the service so that new data source could be more simply integrated, as part of this work I integrated the MoneyGram agent locator service as a new data source, including more integration tests.

Prepaid Card project

The Prepaid Card project was designed to allow Ukash users to buy a pre-paid credit card, that they can then top-up with Ukash vouchers.

I added new functionality into Ukash API to support the main Ukash.com web-site. As part of this work I helped some of Ukash's permanent staff get up to speed with TDD (Test Driven Development), and how the new parts of the web-service worked.

The other piece of work that I did was to incorporate user card data into user profiles in the UMS Admin system. This was achieved by setting up a new action that returns the card display data as JSON data. This is requested by the browser via AJAX, and then displayed using a knockout.js template. Additional changes were added to allow management of various aspects of the users cards. Finally I also added three reports, two are SSRS reports: a report that allows the reporting of prepaid card users who have updated their personal details and a transaction report of cards purchased; also a metrics report the shows users journey through the product, this was setup as an SSIS package as the source is a stored procedure.

Wallet (de-anonymisation) project

Ukash and other anonymous electronic money schemes are being pushed out of certain markets (e.g. Germany) due to changes in local legislation. The Wallet project was conceived

as a way of de-anonymising Ukash, by forcing users in those markets where it is required to register and provide proof of identity.

I added more new functionality into Ukash API to support the the Ukash.com web-site, while at the same time helping permanent staff with TDD principles, creating integration tests and the new design architecture introduced into Ukash API in previous projects.

I also added more SSRS reports into UMS Admin to allow tracking of vouchers purchased and spent through users wallets, and added functionality into UMS Admin to administer users wallets.

Swinton Insurance, Manchester

Swinton Insurance is Britains biggest high street insurance broker with over 600 branches.

Information Systems Development is responsible for the development and maintenance of the systems that power Swinton branches, call centres and head office. The Web Advisor team is specifically responsible for the systems used by advisors in the company's call centres, and back end processing across customer facing systems.

Internet systems - https://quote.swinton.co.uk/Internet/Motor

Also internal equivalents. The public facing quote sites are where most of Swinton's new business is generated. All of the Swinton web-sites are mature, being in use for a number of years. During my time at Swinton part of my responsibilities were to maintain the web-site and insert new functionality as required by the business.

Central Pricing Service

The Central Pricing Service (CPS) is a WCF IIS hosted service used by the branch systems during policy renewal. The service is developed in C# and makes extensive use of nHibernate. The system is supported by an ASP.NET MVC 3 administration portal which is used to upload and manage configurations. Additionally there is a WinForm application that is used to test the service when updates are made.

My responsibilities on CPS initially were maintenance of the service and addition of new functionality. These responsibilities have since expanded to my becoming a key technical contact for the service, providing technical support to other up and down stream teams, and technical guidance/architecting for other team members working on the service.

The most recent project I have lead on this solution has been to extend the MI that is recorded within the solution. As part of the process of discounting new quotes are generated, and have specific discounts applied to them. The business has requested that extra information about the quotes generated, and discounts applied to them are recorded. In order to achieve this the CPS service has been heavily refactored, with the Alternative Quote (AQ) section of the process extracted to a new external service. This was done so that the AQ process could be called early in the discounting process, and proceed in parallel until the AQ is required.

Optima

The Optima service is a WCF IIS hosted service used by the internet systems (internal and external) and branch systems during quoting for new business. The service is developed in C# and makes extensive use of nHibernate. The system is supported by an ASP.NET MVC 3 administration portal which is used to upload and manage configurations. Additionally there is a WinForm application that is used to test the service when updates are made.

The service interfaces with an external Credit Reference Agency to retrieve information on the policy proposer and provide discounts. The service is designed to deal with 600,000 requests per day in a maximum time of 100ms, including the request to the Credit Reference Agency.

The project to build Optima had already started when I joined Swinton, however I soon moved to be one of the lead developers on the project. I was the main technical contact for the service being responsible for monitoring it's health and liaising with up and downstream teams if problems occur. Additionally I was the primary contact within the team for development, both maintenance and new features.

Executive Gateway

The Executive Gateway is a new internal application for the advisors that brings together the many disparate systems that they must use, as well as providing new views on to the data held in those systems.

The application consists of an ASP.NET MVC 4 web application that uses a WCF IIS hosted web-service. The application relies heavily on javascript and jquery to provide a compelling user experience.

The whole of the application has been developed using Behavior Driven Development (BDD), with the c# portions developed against nUnit/Rhino Mocks and the javascript portions developed against jasmine.

My role on the Executive Gateway project was as lead developer. It was my job to interpret the technical direction provided by the Technical Designers. I also helped developers who were working with new technologies, and guided them in applying and understanding the best practices to get the best out of the technologies.

Intechnica LLP, Manchester

The RedRooms web application provides functionality that replaces several of Bruntwood's core systems:

- Meeting room search and booking functionality.
- Meeting equipment management functionality.
- Meeting catering booking functionality.
- Management of irregular meeting space.
- Setting of targets for meeting room lets based on time and financial targets.

Technology

The RedRooms web application is an n-Tier ASP.NET MVC3 application, with data stored in a SQL Server 2008 database.

Data access is done using Entity Framework 4 with POCO (Plain Old C# Objects) objects. The data access layer is split into two projects. The first project defines contains the Entity Framework, the entity models and interface definitions of the repositories used to access the database. The implementation of the data access context and repositories is then in the second project.

Complex processing for the application is carried out by a set of services, these are laid out in a similar fashion to the data access layer, i.e. one project contains interface definitions for the services and model definitions for communication between the web-layer and the services, and the service implementations are in a second project.

The main web application is defined in a fifth application, this contains the controllers, views and attributes and extensions specific to this project. The controllers have two responsibilities:

- The request and aggregation of data to be displayed based on user inputs.
- The collection and validation of incoming data to be processed by services.

The views within the application are purely limited to displaying data to the user. There is some limited processing of lists, etc, in the views, but this is kept to an absolute minimum.

Development methodology

The application has been built using a Test Driven Development (TDD) approach. This has allowed up to four developers to work on the project at anyone time, whilst retaining confidence that in building one portion of the application we weren't breaking another portion.

Responsibilities

On joining Intechnica I took on the lead developer for the Bruntwood RedRooms project. My brief was to decide upon, implement and document the technical landscape of the project; decide upon and implement the technical standards of the project; and train the other members of the development team in the technologies and methods used in the project.

With the support of management I was able to introduce the following technologies:

- .NET 4
- ASP.NET MVC3
- SQL Server 2008
- NUnit
- Rhino Mocks

The principal technical standards of the project are enforced by using StyleCop and FxCop at compile time with all rules turned on. On the developers machine the debug build treats all errors as warnings, allowing the developer to prototype changes to code quickly. However on the continuous integration server all violations of the rules are treated as errors. Developers must not suppress StyleCop errors, instead fixing all errors. However FxCop errors can be suppressed with my agreement.

In addition I was given the time to prove that using TDD would provide the necessary level of comfort with a constantly changing team of developers. (Intechnica were unable to consistently commit more than one developer to the project.)

Intechnica already made some use of TeamCity to do nightly builds of software, I was able to help move this use on to do full continuous integration. As part of this I had to create and maintain an MS Build script that was able to build the whole of the application and apply transformations to the web.config.

2ergo Ltd., Salford Quays

2ergo is one of the UK’s leading mobile marketing and business solutions providers. They provide the Guardians iPhone app and Fox’s Business News and FIFA World Cup iPhone Apps.

The Secure Connect platform provides users with a simple means of designing and deploying J2ME applications that communicate with the core Secure Connect platform via encrypted SMS message. The core Secure Connect platform is a set of Windows services written in C# on .NET 2 that communicate using a mixture of MSMQ and HTTP.

As the lead developer on the project, my duties were:

- Testing and interviewing Indian based developers to expand the Secure Connect team.
- Documenting the Secure Connect platform to allow upgrade on to new Hardware, and allow sale to external customers.
- Design and development of new applications that make use of the platform.

Converting build process from NAnt to MSBuild and implementing Continuous Integration using CruiseControl.NET.

PricewaterhouseCoopers LLP (PwC), UK IT Design and Build (D&B), Manchester

I worked for the largest of the 'Big 4' professional services firms as a senior/lead (Manager grade) developer.

Pensions Bridge – earliest project

Pensions Bridge is a web-based C# – ASP.NET application that allows the Pensions Management Consultancy (part of the Tax Line of Service) to collect and maintain accurate data from providers of software and services to Pensions industry.

The web-site and associated data access layer were developed in C# ASP.NET. Access to the SQL Server 2000 database is done through a combination of views and stored procedures via ADO.NET to ensure the consistency and security of the application data.

Data Analyser v5 and v6

Data Analyser is a desktop based application that imports data from a variety of sources (Microsoft Access and Excel, and CSV files) in to a Microsoft SQL Server Express database. The software then allows the user to mine the data in a graphical and intuitive manner. Data Analyser is owned by the Tax Line of Service, and is actively used to support and sell tax investigation services.

I was responsible for architecting the application design and lead the development of version five, the first version in C# .NET (previous versions were in Visual Basic 6 and used a Microsoft Access back end database). The application was built using Microsoft Smart Client Software Factory (SCSF) and Microsoft SQL Server Express.

I was also responsible for architecting the application design and lead the development of version six. This new version provides the ability to host large data sets within the PwC infrastructure. The hosted part of the application was developed in C# ASP.NET and connects to a SQL Server 2005 database.

HMRC Online Login Browser Helper Object (BHO) for IE6/7, and supporting web-site/service

The HMRC Online Login application consists of:

- a Browser Helper Object (BHO) that plugs into IE6/7 and provides shared login information securely to users;
- a web-site used to administer the logins, the pages they may be used on, and the users who may use them;
- and a web-service for the BHO to communicate with the central database.

Working from a bare business requirement (we need to provide users login information for the HMRC web-site, but the users must not know the login information), I designed and architected a workable solution that met customer and risk management requirements.

The BHO is written in C++. The supporting web site and web service are written in C# using .NET Framework v2 and SQL Server 2005. Automated unit and integration tests have been written using the Microsoft Unit Testing Framework and RhinoMocks to help document the code in the web site and service, and provide certainty that software continues to work in future projects. The project to upgrade the tools was run using agile principles.

CCH PerTax.NET tools integration

PwC use a third party product (CCH PerTax.NET) to manage tax information for certain clients. The supplier has recently upgraded this product to use the .NET framework.

I investigated the existing internally built support tools and then upgraded them to work with new software. I additionally documented the new versions of the software so that they can be supported by an off shore team.

The tools are written using .NET Framework 2 and 3.5 in C#. Additionally they make extensive use of XML, including XPath, XSLT (including extension objects) and some LINQ for XML.

Automated unit and integration tests have been written using the Microsoft Unit Testing Framework and RhinoMocks to help document the code, and provide certainty that software continues to work in future projects. The project to upgrade the tools was run using agile principles.

Additional activities

I lead the .NET developers Community of Interest (CoI) within D&B from its inception in March 2006 to July 2009. The aim of the CoI is to improve the quality of the solutions produced within D&B by promoting knowledge exchange and retention within the developer pool.

As CoI leader has attended Microsoft Tech Ed in the USA twice (June 2006 and 2008) and subsequently presented sessions on new technologies to my colleagues.

Radius Computer Services Ltd, Altrincham

Radius Computer Services was a supplier of payment processing systems to local government, and I worked for them in a Senior/Lead developer role.

RadiusICON was an industry leading payments processing system that included desktop (MFC C++), web (ASP and VB6) and telephone (VB6) payments modules. My work included developing various modules, and designing the technical aspects of the web and telephone payments systems.