Andrea Falconi

programmer

andrea.falconi@gmail.com

Profile

I am a programmer working as a consultant on the architecture, analysis, and design of large-scale software systems. I specialise in the application of mathematical techniques (e.g. algebraic specifications, functional programming) to the specification, construction, and verification of dependable, high quality computer programs. Besides coding, my duties usually entail: strategic technical thinking, design of effective architectures and frameworks, driving product development, definition of technical standards, technology selection, and mentoring development teams.

Education

I studied towards the present-day equivalent of a MSc/PhD in pure mathematics, but had to put aside my studies to start working; however, advanced knowledge of mathematics made it possible to teach myself computer science and how to program with relative ease.

Experience

I have some 15 years' experience in software development and have worked on large projects in different domains such as earth-observation, bio-informatics, medical and business systems. I also have extensive experience in open-source. I have worked with most of the mainstream technologies and platforms—SOA, J2EE, .NET, Oracle, SQL Server, etc.

Employment Record

2005 - Present *Senior Developer/Architect at Dynamic Visual Technologies*

dvt.co.za

Over the years, I have been deployed at several of DVT's local and international clients to implement specialised components as well as to consult on the development of financial & trading platforms, retail & payments systems, and GIS. DVT's clients I have worked for include: Woolworths (woolworths.co.za), ACI (aciworldwide.com), Truworths (truworths.co.za), Allan Gray (allangray.co.za), Visa/Fundamo (fundamo.com), IMQS (imqs.co.za), Achievement Awards Group (awards.co.za), Direct Axis (directaxis.co.za).

2002 - 2005 *Research Associate/Programmer at the University of Dundee*

dundee.ac.uk

I worked on distributed and parallel algorithms for the storage, management, visualisation, and analysis of digital microscope data and metadata as part of the Open Microscopy Environment initiative (openmicroscopy.org)—a joint project amongst the universities of Dundee, Harvard, MIT, Wisconsin-Madison, and NIA Baltimore.

Before 2002

The above summarises the last 10 years. Before then I developed distributed systems for ESA (European Space Agency) and Melograno; I also had several experiences as a Web developer.

Skills & Technologies

Software technology is the result of the application of scientific knowledge (Computer Science/Mathematics) to the solution of concrete problems and is produced by a disciplined approach based on software engineering principles. A sound grasp of both the scientific knowledge and engineering principles used to produce software enables me to quickly become proficient with new technologies (e.g. learning a new programming language or framework) as well as to play in the technology-making arena (e.g. *creating* a new language or framework).

Below is a summary of my programming skills and the technologies I have been working with.

Specification, Construction, and Verification

Abstract algebra, discrete mathematics, mathematical logic; automata, formal languages, compilers, interpreters; algorithms & data structures; concurrent, parallel, and distributed programming; object-oriented, functional, and formal methods of analysis & design; architectural and design patterns, frameworks; formal verification; regression/integration/performance/system testing; debugging & profiling.

Technologies: **C**, **C++**, **C#**, **F#**, **Java**, **Scala**, **Haskell**, **SQL**, **UML**, **XML**, **XSLT**, **HTML**, **CSS**, **Perl**, **Ruby**, **JavaScript**, **ActionScript**, **xUnit/mock** frameworks, **QuickCheck**, several performance benchmarks.

Operating Systems and Networking

Hardware architectures; processes and threads, IPC, scheduling, virtual memory, I/O, file systems; network layers, protocols, addressing, routing.

Technologies: Knowledge of **Unix** and **Windows** platforms, system libraries, shells, **Linux system administration**; **Internet protocols** (IP, TCP, UDP, HTTP, etc.) and their C/C#/Java/Haskell APIs.

Databases

Relational algebra and calculus; entity-relationship modelling; normalisation; indexing; transactions.

Technologies: **Oracle**, **SQL Server**, **PostgreSQL**.

Distributed Systems

Client-server & peer-to-peer models, message-oriented middleware & distributed objects, n-tier architectures, directory & discovery services, security, concurrency control, failure handling, replication, performance, scalability, transparency.

Technologies: **J2EE**, **.NET**, **SOA**, **Web Services**, **REST**, **Ajax**, **CGI**, several application servers and Web frameworks, **CORBA**, **ICE**.

Software Process

Waterfall model, rapid application development, iterative & incremental methods; software configuration management.

Technologies: **Agile**, **TDD**, **XP**, **RUP**; most of the commercial and open-source version control systems; several build, continuous integration, requirements & bug tracking systems.

Samples

Due to contractual agreements (confidentiality, copyright, etc.), I cannot disclose any technical information regarding the projects that I have been working on for the past 8 years. However, I do have some samples on GitHub that you could use to evaluate my skills and asses how your organisation may benefit from them:

- **Introducing Mathematical Methods** —
  https://github.com/lambdacat/samples/raw/master/intro-math-methods.pdf

- **ISO 8583 Validation** —
  https://github.com/lambdacat/samples/raw/master/iso8583-validation.pdf

- **A Taste of Algebra of Functions** —
  https://github.com/lambdacat/samples/raw/master/algebra-of-functions.pdf

Please read below for the context in which these samples should be examined and for a summary of their contents.

Although I am well-versed in mainstream software development (e.g. object orientation, SOA, etc.), I believe my most valuable expertise lies in the application of mathematical techniques to the engineering of dependable, high quality, yet cost-effective systems. Indeed, every so often, I give talks on the topic of mathematics in software engineering. A common thread in these talks is to show how using mathematical techniques to develop software can improve over the current state of the art, leading to higher quality products in a fraction of the time.

Introducing Mathematical Methods summarises my prototypical talk of this type for a non-technical audience; it is essentially a business case for the adoption of mathematics in software engineering. In it, you will find a case study in which are quantified the benefits of the mathematical approach over the mainstream, object-oriented one. Following this talk, there is usually a technical session in which I demonstrate development using algebraic techniques.
ISO 8583 Validation is the transcript of one of those sessions, showcasing a validation framework for financial transactions.

I have also run several workshops to teach algebraic methods of software development.
A Taste of Algebra of Functions contains material that I use during the first of a series of workshops; it presents several problems involving central themes in programming (such as abstraction, reuse, correctness, performance) and their mathematical solutions along with the corresponding implementation in the Haskell functional programming language. It is a good example of how to specify, construct, and verify dependable, high quality computer programs.