

Curriculum Vitæ

Education

- 2006** Engineering degree in Computer Science. *Universidad Nacional de Educación a Distancia*.
1997 Ph.D. in Theoretical Physics (*Spherical Gravitational Wave Detectors*). University of Barcelona.
1992 Master degree in Theoretical Physics. University of Barcelona.

Professional Experience

Jul 07 – Jul 08 Software Engineer, Google

Some engineers at Google were reading my *programming musings* blog and liked it enough to contact me. After a bunch of interviews, I had an offer on the table to join Google's Zürich office which was difficult, if not impossible, to refuse.

I had a great time working as a software engineer at Google. My work there revolved around the technologies underlying web search—that is, I became pretty well acquainted with technologies such as mapreduce, bigtable or the google file system—with an additional focus on quality issues.

Jun 05 – Jun 07 Software Engineer, Institute of Space Studies of Catalonia ([ICE/IEEC](#))

Project: [LISA Pathfinder](#) (NASA/ESA), a satellite-based experiment to assess the feasibility of [LISA](#), a space-based gravitational wave detector.

Responsibilities: Design and implementation of two embedded control modules responsible of coordinating and monitoring the scientific instrumentation in the spaceship, as well as data acquisition. In addition, I've been in charge of writing and deployment of QA infrastructure (coding and testing standards, source control management tools, etc.).

Technologies: One of the modules runs on the bare metal, so it required the implementation of a small (soft) real-time kernel and drivers. The second one, having hard real-time constraints, is being implemented on top of [RTEMS](#). Our hardware includes a [MIL-STD-1553](#) communications bus and controller, a RISC processor (ERC32), and EDAC-protected memory chips.

Languages & tools: C for the main modules, Python and Emacs for test software and glue. *Cantata* for unit testing and coverage. The host environment is centered around Emacs running on Debian. Doxygen and \LaTeX (with custom macros) for documentation. Bugzilla and GNU Arch for issue tracking and SCM. MoinMoin as knowledge-base and general coordination tool.

For more details, see [this blog entry](#) and my recent short review [Mission-critical software development in LISA Pathfinder](#).

Sep 03 – Sep 05 Associate Professor of Computer Science, Autonomous University of Barcelona

Teaching

Programming methods and technologies, full-year sophomore course.

Computer Networks, half-year graduate course.

Short (30–40 h) courses on GNU/Linux Server Administration and Java Cryptography.

In addition, I used WikiWikis as a teaching aid for the first time in that university. They are nowadays quite popular among other professors of the AUB, even as an evaluation tool.

Professional Experience (continued)

Research on Mobile Agents and distributed systems, with a focus on security issues (see for instance [this AAMAS 04 paper](#)) and continuation-based code migration. [This paper](#), written in collaboration with Apache Software Foundation's Torsten Curdt, gives a good overview of our work on the latter.

Teaching was an interesting experience, but the lack of interest of many of the students was a bit disheartening. After a couple of years I felt ready again to plunge into software development, and the offer to participate in LISA Pathfinder was anything but easy to refuse.

Sep 01 – Aug 03

Software Responsible, [Scytl Online World Security](#)

Responsibilities: I was among the initial employees of this startup, focused on secure e-voting systems, and, as the senior software engineer, was in charge of organising and coordinating its whole development team (5-8 people). This covered the full gamut from development and QA standars to setting up our environment.

Project: As a developer, I was the technical leader of [Pnyx](#), the company's main product. It consisted on a set of distributed, communicating processes, highly portable (Linux, FreeBSD and Windows were supported) and with stringent reliability, security and QOS requirements.

Technologies: State of the art cryptograhic protocols (built upon public key cryptography) were implemented with C++ as the main implementation language. Interprocess communication and network architectural patterns based on Schmidt's book and libraries (ACE) conformed the base of our development (together with Openssl). We also took the chance to learn and use template metaprogramming and get acquainted with the excellent *boost* library. Stroustroup, Alexandrescu, Sutter and Meyers were our references by then.

Languages & tools: C++, as mentioned, but also Java for early prototypes and client-side applets. Our main platform was GNU/Linux/Emacs and, our scripting language, Perl. \LaTeX for documentation, Mantis as BTS, CVS for SCM and Umod as our knowledge-base.

See [this paper I co-authored](#) for more details on the kind of problems we were trying to solve.

Apr 00 – Sep 01

Software Engineer/Research Manager, [ISOCO](#)

Responsibilities: I joined ISOCO when it was a very young startup looking for investors. As a result, one of my first responsibilities was to help explaining venture capitalists our ideas on my area of *expertise*, namely, personalization. Actually, I was *learning* about it during this process (and far afterwards). ISOCO was an spin-off of a university institute devoted to AI research, and the initial ideas we had were to apply some AI techniques to webapps and other on-line services. Once we got our seven-digits investment (those were the times of the dot-com bubble), I went back to actual software development.

Project: I spent all the rest of my time at ISOCO devoted (first as a developer and eventually as its technical leader) to our personalisation project, *Alice*. An early prototype (for an American assurance company) consisted on a Bayesian decision tree applied to discovering the most adequate assurance policy for a given customer (based on her age, sex, income, and so on). Afterwards, we moved to providing a generic personalisation service for websites based on a forward-chaining rule engine. The main idea was to attach attributes to every clickable item in the site and follow the users navigation through the site's page. From that we obtained a profile and produced life recommendations via our rule engine.

Technologies: All our development was based on the Java standard and, to a lesser extent, Enterprise platform. We used servlets on the server-side and JSPs and tag libraries for the client-side. Swing was the base for our graphical database and rule editors. Being generic, the database attributes and relationships implementations were heavily based on metadata structures which were quite similar to the mechanisms one encounters these days in Apple's Core Data libraries. AI techniques included, as mentioned, Bayesian networks, rule engines (we used initially a home-brewed implementation based on CLIPS and updated later on to a commercial solution) and a bit of data mining.

Professional Experience (continued)

The things I loved about ISOCO were its highly talented staff and their hacker culture. I learned a lot from them, and people were enthusiastic about making a dent in the web applications space. Unfortunately, when the big money came and the investors got an iron hand on the board of directors, this hacker culture was progressively diluted. After completing the first version of *Alice*, the project was closed (we wanted to add automatic learning capabilities for version 2) and I decided to look for greener pastures. Scytl was founded by a bunch of ISOCO trying to recover its initial, refreshing spirit.

Jul 99 – Apr 00

Software Responsible, [Indra Espacio](#)

Responsibilities: After completing the development and deployment of *Arcanet* (my previous project at Indra, see below), I was assigned the role of Software Responsible for the development of the Earth station of ESA's [EGNOS](#) project (the [Galileo](#) precursor). As such, I wrote the development and coding standards for the project, wrote the Software Requirements and the Software Architecture, leaving the company after the project's CDR was successfully closed. I also played a very active role in reviewing and selecting the COTS to be used in the project.

Project: We were in charge of developing the Earth stations receiving the GPS signals and distributing them to central computers in the network. The software part was classified as safety critical (level B) by ESA and, therefore, we were under strict development standards and processes based on Civil Aviation's DO-178B and ESA's PSS 06. Our software ran on a commercial RTOS, with soft real time requirements.

Technologies: Most of all, in this project I learned about formal processes and QA methods in the context of a safety critical application development. We received training by [TekSci](#) on this kind of development, and, specially on a [DO-178B-based software process](#).

Tools: Since I only worked on the project until the completion of the architectural design, the tools I used were only for requirements gathering (DOORS) and design (Aonix's Software Through Pictures). But I also reviewed candidates for our RTOS (I chose Lynx) and unit testing and coverage (Cantata).

Although EGNOS was a pretty interesting and challenging project, it involved an enormous documentation and QA practices overhead. I realised I was spending most of my time writing and reviewing documentation and, due to my being in charge of coordinating the software development, the situation was not likely to change. I love programming and wanted to keep learning about programming. In addition, ISOCO offered a way out the rigid corporate culture of Indra. So there I went.

Oct 97 – Jul 99

Software Engineer, [Indra Espacio](#)

Responsibilities: *Arcanet* was my first serious project as a developer. It consisted of a satellite-based telephony system deployed by EUTELSAT, our contractor. A series of central stations connected with regular telephone trunk lines provided access to voice and data services to mobile, portable remote terminals. EUTELSAT satellites mediated in the communication between central stations and remote terminals, using CDMA channels. Indra developed both the hardware and the software for EUTELSAT. I was in charge of developing all the software controlling the central stations. I also took a very active role in the deployment of the final system, spending about forty days at Larnaca's Satellite Station in Cyprus.

Project: Each central station used two redundant computers running Windows NT. The software consisted on a set of intercommunicating, redundant processes providing call processing, PSTN connection, and HW/SW monitoring and control services.

Technologies: Our solution was based on multithreaded processes communicating via UDP messages in a local network. QOS and availability requirements implied strong separation of concerns and special processes dedicated to monitoring (via, e.g., heartbeats) the whole station's software. Some years after finishing my implementation I would realise that I had reinvented many of well known network architectural patterns, creating essentially a poor-man's version of the [ACE library](#) (which we used extensively at Scytl).

Professional Experience (continued)

Languages & tools: C++ was the main language, with Visual C++ 4.5 and 5. SQL Server was used as our database, with much of the business logic for satellite resource management and customer accounting programmed as stored procedures.

Although I was the only developer of the programs mentioned above, I learned a lot during this project by actually hands-on work and, of course, lots of reading (cover-to-cover C/C++ Users Journal was my favourite those days). Participating in the acceptance tests run by our contractor and in the deployment of our hardware and software in a real satellite station, also only tangentially related to software development, was also quite enriching.

Non-professional Experience

During all these years as a professional programmer, I've devoted much of my free time to learn more about computers and computer programming, specially in those areas that were absent in my daytime jobs. Thus, I'm a long time user of GNU/Linux systems and always keep a Debian machine near me. I have also developed a passion for everything related to programming languages, and, during the last two years, have discovered the beauty and power of Macs and OS X.

GNU MDK

[GNU MIX Development Kit](#) has been my vehicle to learn about Free Software tools and processes, as I kept adding new features and using new development tools in each release.

MDK is a development environment emulating Donald Knuth's MIX computer, including a virtual machine, a bytecode compiler, a debugger, graphical interfaces and (my favourite feature) extensibility via Guile.

MDK became an official GNU package by the end of 2000 and its Users Manual (which I authored) was published [in printed form](#) by the GNU Press in 2003.

Conjure

[Conjure](#) is (was) meant to be a Make replacement written in Scheme, in the spirit of Scons or Rake.

Besides for being the most beautiful language ever, Scheme is famous for having lots of incompatible implementations. Since I wanted Conjure to be portable among implementations, work began with providing a compatible mechanism for reusable code (and a meta-module system). A friend of mine, [Andreas Rottman](#), joined this effort, and [Spells](#) was born as a byproduct.

In parallel, I developed an alpha version of Conjure for Scheme48. Unfortunately, after finishing this proof-of-concept version, we have had no time to push the project forward. But, at any rate, it's the most significant amount of Scheme code I've ever written.

Snippets

Other contributions to existing projects:

- I wrote the original version of the *diagram browser* included in [Dia](#).
- Various contributions to Tom Lord's [Hackerlab library](#).
- Contributions to the Scheme interpreters/compilers [Pika](#) and [Guile](#).
- A handful of Emacs [packages and extensions](#) in Emacs.

Languages

My appreciation of (and eventually passion for) functional/dynamically typed languages began while reading SICP and *The little/seasoned Schemer* books. I fell in love with Scheme and, after reading *On Lisp* and playing with Slime, with Common Lisp. I've written many short programs in Lisp since then, and Conjure was my first serious attempt at doing serious stuff. I've also gained an appreciation for the ML-family of languages, specially Haskell, but I have yet [to learn it well](#). I have also certain familiarity with more theoretical aspects of programming languages, [including category theory](#).

Some of my experiences and thoughts on this field are covered in my [Programming Musings](#) blog (the links above point to some of my entries in there).

Non-professional Experience (continued)

Mac OS X

I discovered Macs and OS X around 2005, and during a couple of years it became my default system. During that time, I devoted almost all of my computer-flavoured free time to learn seriously about Cocoa and related technologies. I was also developing a little application that could be released as shareware (in the Mac ISV tradition); the point being to complete a full application so that i felt qualified to jump into a real world Mac development should the opportunity arise.

Other Interests

Physics

Physics was during many years my great passion, and I still keep a keen interest in it. I try to keep current in the fields where I once had some expertise (General Relativity, foundations of Quantum Mechanics) and maintain a discontinuous blog called [Physics Musings](#).

My current employer is a governmental institution whose main activity is research (but my main job is software development). That led me at first to daydream and play with the idea of doing some research again (as [this overoptimistic blog entry](#) reflects). But I've realised by now that real, quality research is not something that you can resume whimsically, and that currently I'm much more a software engineer and programmer than a physicist.

Books

I'm addicted to reading and collecting books. I love good literature, from Joyce and Cervantes to Stephenson or Dick. I also read as much as I can on philosophy and cognitive science, and good science writers like Hofstadter or Dawkins. I also enjoy writing (very) short fiction, and still hope to publish a novel when I'm sixty-four.

Karate-do

In my non-existing free time, I still try hard to go to the gym at least once a week, to practice a bit of karate and pick-up a little Japanese culture and language in the process.