Tom Ritchford

- http://github.com/rec
- tom.ritchford@gmail.com
- +31 64 121 2749
- Amsterdam, Netherlands

I specialize in rapid development of highly reliable, performant, scalable, minimal, clear and maintainable solutions to difficult problems.

Decades of experience! A plethora of projects taken from conception to completion, production, packaging and distribution.

Expert in Python, C++, solid in Java, C and JS, conversant in many others.

Skills

- backend
- search
- fintech: option modeling, ledgers, position management
- big data
- distributed systems
- audio and DSP
- real-time control
- and more

"Everything should be made as simple as possible, but no simpler" -various

Employment Highlights

Lead Programmer, SuperDuperDB (May 2023 -- Sep 2023)

https://github.com/SuperDuperDB/superduperdb

SuperDuperDB is not a database, but a Python system integrating existing databases with AI tools like vector search and LLMs.

I was tasked with making the working Python codebase "professional" so to be released to the public as a library.

And this I did, by writing many `pytest` tests, adding practically complete typing enforced with `mypy`, refactoring and renaming and removing cruft, and of course writing reams of readable class, method and function documentation.

I also got to write some design documents, really my favorite part: for *structured logging, monitoring, and journaling*, which had a small but useful API, and allowed repeatable computation with fairly marginal extra effort; and a design for a *fully typed REST server* (including a tiny demo!) with an automatically generated OpenAPI specification, instead of the existing ad hoc REST server, using FastAPI and Pydantic

CTO, Engora (April 2021 -- Feb 2023)

Engora was an innovative search engine for mechanical engineering parts.

The founder created a good demo, and then raised money through crowdfunding. I came in some months after as CTO.

As the CTO I had my hands in everything, but here are the bits I wrote all of (Python, PostgreSQL, SqlAlchemy):

- A *parts crawler* over two dozen disparate websites totalling almost a million parts, carefully rate-limited, first harvested directly, then proxied, finally using ScraperAPI's fancy new asynchronous proxy.
- A PostgreSQL *parts database* with key information from each parts page: I wrote a small database quickly, and then four months later, I rewrote it entirely when I knew what I was doing.
- A *data store* based on S3, replicated over multiple providers and with an incremental offsite "physical" backup stream; and on top of that, a *data resource management system*, for convenient replication of resources, and projects containing multiple, reproducible resources, including PostgreSLQ databases, directories and sharded files.
- A neat little proprietary *memory-mapped index* for direct searching and retrieval, and Whoosh for text searching.
- A Flask *web server* (using nginx/gunicorn in production) and a couple of Dockers supporting all of these.
- Deployment, configuration files and variables, monitoring variables, logging, user interaction journaling, and other unsexy but satisfying details.
- "Practically complete" test coverage of everything
- And to run all of those, a tidy `typer` CLI named `engora`, with over two dozen commands and subcommands, hundreds of flags and "practically complete" documentation, used every day by almost everyone in the company.

Lead developer on BiblioPixel, Maniacal Labs (2016-2019)

Maniacal Lab's BiblioPixel was a popular lighting control program written in Python that controlled LEDs in strips, matrices, cubes and other layouts, as well as other lighting systems such as the Philips Hue and DMX.

I rewrote it from the ground up, with a REST server for pixel and higher-level control, both code and data plug-ins, animators including video feedback with an IIR filter, and a new data model using `numpy` arrays, leading to very roughly a 30x speedup with perfect backwards compatibility.

Mostly Python, some Cython and C++. (I'd use pybind11 instead of Cython if I had to do it again.)

Senior software engineer at Ripple (2014-2016)

Ripple is a financial technology firm with its own eponymous cryptocurrency. I worked on their flagship application `rippled`, the complex and complicated C++17 crypto-ledger that implements their XRP cryptocurrency, on the ledger code, on deployment, debugging, devops, build and monitoring, mostly in C++ with some Python.

CTO, World Wide Woodshed (2009-2014)

I had always wanted to write a complete desktop audio application!

World Wide Woodshed's SlowGold was a leader in music practice software from the 1990s. I bought half the tiny company, and was the sole developer for a brand-new product in C++, with high-quality audio, subtle and intuitive editing tools, and little details like three second restart after shutdown.

Software engineer, Google (2004-2009)

I joined Google New York when it was a single floor overlooking Times Square, worked on Google's first question-answering system, the first Music Search, then its short-lived Real Estate search.

This led me to GoogleBase, a database of tens of billions of items planned for millions of users. Leading a tiny and changing team, over two years we built a universal reporting and computation framework I had proposed and designed. It was still in common use years later.

As a reward for this slog, I was privileged to work on GWS, the front end program, written in C++ that generated all Google results pages, for i18n, l10n and translations, and the GWS live experiment framework.

And I interviewed hundreds of engineers, traveling twice to Korea and once to Hungary for this.

I used C++, Java and Python, and the usual string of Google technologies.

Senior software developer, Netomat (2001-2004)

Netomat had an innovative rich media tool to let users and advertisers create and send Netomat "experiences" – little Java applets (it seemed more reasonable at the time) minisites with animation, sound and internal navigation - to users who could edit them within the email itself.

I designed and wrote the animation engine and front-end, most of the animation types and the manual.

Still one of my favorite "neat hacks" ever, I wrote a tool that converted "experiences" right into Java bytecode, for a 40-80% savings in download and memory size.

Skills

- Architecture and high-level design: clean, simple, practical, scale-appropriate, 12-factor
- Brutal, thorough testing and CI
- Python: Flask/SQLAlchemy/Django/FastAPI/Pydantic, numpy, Cython, real-time, packaging, typing!, and more...
- C/C++: modern C++11-20, STL, DSP, concurrency, Juce, Boost, real-time, digital audio
- Java: distributed systems, i18n/l10n
- Considerable Javascript, strong Linux, Bash scripting
- Data analysis and retrieval: clustering, search and indexing, data pipelines, S3, MapReduce, log analysis
- PostgreSQL database design, use and some admin
- Strong Git (I wrote this: https://github.com/rec/gitz)
- Practical DevOps: sysadmin "classic", deployment/release/integration, monitoring and logging
- Globalization: Internationalization, localization, translation, Unicode and encodings
- Performance optimization
- Fintech: ledger systems, option models
- Real-time systems: digital audio and DSP, lighting control systems, MIDI
- Tool building: see my tools dashboard at https://github.com/rec

Education

I have a B.Sc. with First Class Honours in Mathematics from Carleton University, Canada.