Tim van Deurzen

SYSTEMS ENGINEER • C++ AND GO • AMSTERDAM, NL

tim@kompiler.org
+31 6 117 617 49
github.com/vdeurzen

Thirteen years of systems engineering on Linux, in C++ and Go, with experience in columnar analysis engines and query performance, concurrent data pipelines, infrastructure and internal tool design. Long-standing interest in programming languages and compilers, academically through an M.Sc. thesis on partial evaluation for automatic JIT generation and professionally through language embedding (SpiderMonkey and LuaJIT) and API design. Contributed to GCC and to the C++ standard, and follow the language closely through volunteer work at CppCon, ACCU on Sea and C++Online. I enjoy learning new languages, algorithms and data structures, and diving deep to build my understanding. Looking for systems work in or adjacent to open source.

EXPERIENCE

Apr 2023 – Jun 2026

Senior Software Engineer @ Liquid IT

Financial forecasting and reporting platform.

Go · PostgreSQL · NATS · Kubernetes · Terraform

- **Integration architecture.** Designed the integration data architecture and built three accounting synchronisation services in Go.
- **Task scheduling.** Designed the asynchronous task module the synchronisation services are built on. Dependencies are declared through a fluent DAG API and scheduled with Kahn's algorithm, so independent tasks run concurrently in goroutines.
- **Service architecture.** Moved the analysis system from MongoDB to PostgreSQL and from TypeScript to Go, and migrated key services onto NATS so tasks and events persist and can be retried. ConnectRPC and protobuf for inter-service streaming, generating both Go and TypeScript definitions.
- **Infrastructure as code.** Led the move to Terraform, Kubernetes and AWS, running the migration off a hand-managed VM and leaving an automated deployment pipeline behind.
- **Observability and monitoring.** Deployed Prometheus, Grafana and Loki as our observability stack, providing quick insights into production issues.
- **Ledger classification.** Implemented a Naive Bayes classifier mapping customer ledgers into the RGS taxonomy, after comparing trees of classifiers, SVM and hierarchical arrangements. Onboarding went from several meetings to one.
- **Reporting on Postgres.** Built an accounts receivable and payable dashboard on Postgres, from the data model through to the frontend components, development supported through agentic workflows.
- **Agentic development.** Helped architect the platform's LLM integration and agent tooling, and followed that with a closer study of the practice itself: spec-driven development, harness and skill design, and agentic review with Claude and CodeRabbit.

Aug 2019 – present

Owner, Independent Consultant @ Eolas Engineering

Go systems consulting and technical editing for Manning.

Go · Kafka · protobuf

- **Event processing framework.** Designed a generic asynchronous processing framework in Go for a client, integrating Kafka and protobuf, with support for external offset storage.
- **Technical editing.** Editorial credits on four Manning Publications titles: *Learn C++ by Example* (technical editor), *100 Go Mistakes and How to Avoid Them* (technical development editor), and *Go by Example* and *Algorithms and Data Structures for Massive Datasets* (technical proofreader).

Apr 2018 – May 2019

Senior Software Engineer @ Blockport

Cryptocurrency exchange. Service-oriented wallet infrastructure in C++ and Go.

C++17 · Go · libbitcoin · Crypto++ · ZeroMQ

- **Threshold key custody.** Split the master key across multiple holders with Shamir's secret sharing using Crypto++, so reconstruction required a quorum to agree and no single person could recover it alone.
- **Key derivation.** Implemented BIP-32, BIP-39 and BIP-44 derivation and management in both C++ and Go, with a CLI for managing the key hierarchy, built on top of libbitcoin's established components rather than hand-rolled primitives. Efficiently tracked thousands of wallet addresses.
- **Deposit monitoring.** Tracked deposits and traced payments in C++17, wrapping libbitcoin for the peer-to-peer layer, with deposit consolidation. Used Conan for dependencies, writing recipes for the libraries that had none. Built an equivalent service in Go for ETH and ERC-20 deposits.
- **Compliance monitoring.** A separate Go service consuming the Bitcoin ZMQ interface, tracking block lineage in a map-based structure to follow the correct chain.

Apr 2013 – Apr 2018 • **Senior Software Engineer** @ GfK Business Applications

Hyper8, a columnar indexing and analysis engine: a C++11 library behind a PHP interface.

C++11 PHP SpiderMonkey MongoDB Lua

- **Bitset-level evaluation.** A client's analysis was a large boolean formula that the interpreter evaluated row by row. Rewriting it as AND and OR operations directly on the underlying bitsets reduced the runtime from more than 48 hours to seconds. Generalized the technique and exposed it to the interpreter for all users.
- **On-demand updates.** The engine was columnar, but increasingly used transactionally. Helped design and implement durable status column updates, including the JSONL journal format, from development to production.
- **Ingestion pipeline.** Replaced error-prone file concatenation and `cut` with a MongoDB pipeline, following an evaluation of CouchDB, Cassandra, Couchbase and MonetDB.
- **Embedded languages.** Maintained the embedded SpiderMonkey engine in the PHP interface, and prototyped embedding Lua and JavaScript inside the engine itself, with the aim of retiring the PHP wrapper and running the system as a server.
- **Query parser.** Maintained the parser for the in-house SQL dialect by hand, after the generator infrastructure it was built with was lost.
- Also drove the initial Dockerization, owned production deployments, and served as Scrum master for the data team.

OPEN SOURCE WORK

2019 – 2020 • **CsCrypto** @ CopperSpice

- Implemented the first version of the abstraction over cryptographic libraries, with OpenSSL and Botan as the initial backends and modern C++17 API design throughout. github.com/copperspice/cs_crypto

2019 • `std::function_ref` **(P0792)** @ WG21

- Contributed to the original proposal for a non-owning callable reference and am acknowledged in the paper. Adopted for C++26.

2018 – 2019 • **Spaceship operator** @ GCC

- Implemented the initial parser changes for three-way comparison (P0515), working alongside GCC developers that I met at CppCon and the Belfast ISO C++ meeting in 2019.

2010 – 2012 • **Maintainer** @ CMix II

- Maintained CMix II, the partial evaluator developed at DIKU and the central tool in my M.Sc. thesis.

2010 – 2012 • **Runtime specialisation** @ Single Assignment C

- Implemented the runtime specialisation framework in the sac2c compiler, out of the B.Sc. thesis work. sac-home.org

ongoing • **Speaker Liaison and volunteer** @ CppCon, ACCU on Sea, C++Online, ADC, Meeting C++

- Speaker Liaison for CppCon, covering speaker communication and logistics across the programme. Volunteer at ACCU on Sea, C++Online, ADC and Meeting C++. Attended the ISO C++ (WG21) meeting in Belfast, 2019.
- Ongoing research into agentic development techniques, applied to personal projects where relevant.

TECHNICAL

Programming	C++ 11–23 · Go · C · Rust · Python · SQL · Lua
Systems	Linux · Neovim · Git · Docker · Kubernetes · Terraform · AWS · CMake · Conan · gdb · agentic development
Data	Postgres · DuckDB · MongoDB · Redis · Kafka · NATS · protobuf · ConnectRPC
Focus	query engines · JIT and compilers · concurrency · performance analysis
Spoken	Dutch, native · English, fluent · Danish, fluent

EDUCATION

2010 – 2013 • **M.Sc. Computer Science** · University of Copenhagen

- Programming languages and systems, focused on partial evaluation and compiler technology. Thesis, supervised by Robert Glück: *A Case Study of Using Offline Partial Evaluation to Automatically Generate JIT Compilers for Virtual Machines*. Prototyped the derivation against subsets of the JVM and Lua VMs.

2007 – 2010 • **B.Sc. Computer Science** · University of Amsterdam

- Thesis, supervised by Clemens Grelck: a runtime specialisation framework for Single Assignment C, since included in the sac2c compiler.