Andreas Bernhard Scharf

E-Mail: scharf.andreas.b@gmail.com
github-account: https://github.com/Schwarf/
Last updated: 12/2025

Senior **C++/Linux** engineer (7+ yrs) focused on correctness and performance. Owned a certified header-only core library used in **20+ repos**; shipped **2×–17×** speedups with reproducible benchmarks and **CI perf gates**. Strong algorithms background; most impact in **single-threaded** hot paths; expanding hands-on multithreading for low-latency.

Robert Bosch GmbH — Research Engineer → SW Developer → Senior SW Developer → Product Owner (04/2018 – Present)

C++ Engineering & System Architecture (C++11/C++14/C++17 on GCC/Clang)

- Optimized an optional-like type in a certified header-only C++ library via specialized storage → **2×** runtime for trivial types and **5–20%** lower memory on embedded targets; rolled out across **10+ compilers.**
- Added move semantics to fixed-capacity containers (vector/list/map/tuple) → **~10%** runtime gain in release builds (Google Benchmark) across **10+ customer toolchains**
- Built a Docker-based micro-benchmark harness (Python) for GCC/Clang; used Valgrind **instruction counts** for GCC/Clang and reproducible **CPU-cycle measurements** via an internal simulator toolchain; integrated results into CI to prevent regressions.
- Delivered up to **17× speedups** on selected C++ APIs using the benchmark harness by restoring missing inline qualifiers, refactoring call sites, and adding type-constrained function templates.
- Refactored real-time radar modules **(–1,500 LOC, –25% complexity**) by redesigning legacy “iceberg” classes and templates, improving determinism and maintainability in safety-critical code.
- Debugged and stabilized a k-d tree nearest-neighbor implementation used in LiDAR perception; added targeted edge-case unit tests to prevent regressions.
- Resolved static-analysis findings at scale (constexpr, POD init, safer initialization).

Python Tooling & Data Engineering

- **~40× faster** rare-event hunts by building Python pipelines to retrieve/convert/analyze ~1,000 radar sequences.
- **~33% less refactor time** via an automated Python validation toolchain that checks thousands of signals and gates changes during large C++ refactors.
- **~5% maintenance** reduction by scanning **20+** customer repos to identify ~**50** unused public APIs and drive deprecations, shrinking the supported surface area.
- Operated an **Azure SQL metadata store** and integrated Azure DevOps/ML pipelines for automated data delivery and schema management.

Product Owner, cross-BU C++ base-library team *(Sep 2025–Present — concurrent)*

- **Co-initiated and secured** the consolidation of two parallel C++ base-library efforts; presented the plan to the architecture council and CTO Techboard, enabling the formation of a ~**10-engineer cross-BU team.**
- **Authored a 30-page 2026 business plan**, defined the three feature streams, and established roadmap, milestones, and architectural runway.
- **Owned customer relations** as the single point of contact for requirements, prioritization, and expectation management.
- **Transferred performance benchmarking practices** from prior work, ensuring adoption across feature streams.

- **Defined epics, user stories, and review cadences**; aligned stakeholders on scope, responsibilities, and quality standards (API, performance, architecture).
- Continued contributing as **a hands-on developer**, acting as a firefighter across all feature streams to unblock critical paths.

Leadership and Team Impact

- **Drove a customer-centric workflow (without a formal PO role):** set up intake/prioritization, reframed tasks as epics/stories, and delivered initial requests to demonstrate the standard; result: quicker turnaround and consistent developer–customer contact.
- **Established Developer Relations for the base library:** single contact for developer/customer issues; triaged and resolved escalations, **turning a frustrated customer** into a reference user and reducing back-and-forth cycles
- **Cross-team alignment on labeling workflows (LiDAR/Radar/Camera):** initiated and facilitated workshops to harmonize strategies, reducing handoffs and communication overhead and improving delivery predictability.
- **Internal enablement**: delivered intro talks on **Python** and **GoogleTest (gtest)** for junior engineers.
- **Software Project Lead, ASAP Engineering (11/2017–04/2018)**: led a 5-engineer team; introduced agile practices (open comms, team-driven decisions, iterative feedback).

Open-Source contributions

- **NetworKit (C++20) — Selected Contributions**
  - **Left–Right Planarity Test:** Near-linear planarity check for large graphs (merged), follow-up optimization delivering up to **~100× runtime improvement** (merged).
  - **Successive Shortest Path (Min-Cost Flow):** Computation of cost-optimal flows under capacity/supply constraints **(**merged**)**.
  - Full PR history available on GitHub.
- **Gonum (Go)**
  - **Dinic's Max-Flow** algorithm (merged).
  - **Complex Dilogarithm (Li$_2$):** Numerically robust implementation with full test suite (merged). Added performance and benchmark suite with **up to 1000× speedups** (merged).
- **NetworkX (Python)**
  - Bug fixes: Issue #7645, Issue #7796

Earlier Experience

Software engineer - ITK Engineering GmbH, 05/2015 – 10/2017

- Fixed numerical-kernel issues in client C++ code (Fresnel integrals), restoring correctness under real-world inputs
- Evaluated real-time charting/plotting performance on constrained customer hardware (C#/C++), informing tech choices
- Contributed across small projects in C++/C#/Python/Java (tooling and integrations)

Academic Research & Education (List of publications)

Theoretical Physics — KIT (Ph.D., *magna cum laude*) | **KIT** (Diplom / M.Sc. equiv., grade 1.1)
Postdoctoral Research (2008–2015) — SUNY Buffalo & University of Würzburg
Large-scale numerical computations and Monte Carlo simulations (Fortran); peer-reviewed publications; collaboration & supervision

Tools & Languages (applied)

C++17/20 (GCC/Clang), Linux, CMake, GoogleTest, Google Benchmark, Valgrind/Cachegrind, Actions, Python; Go