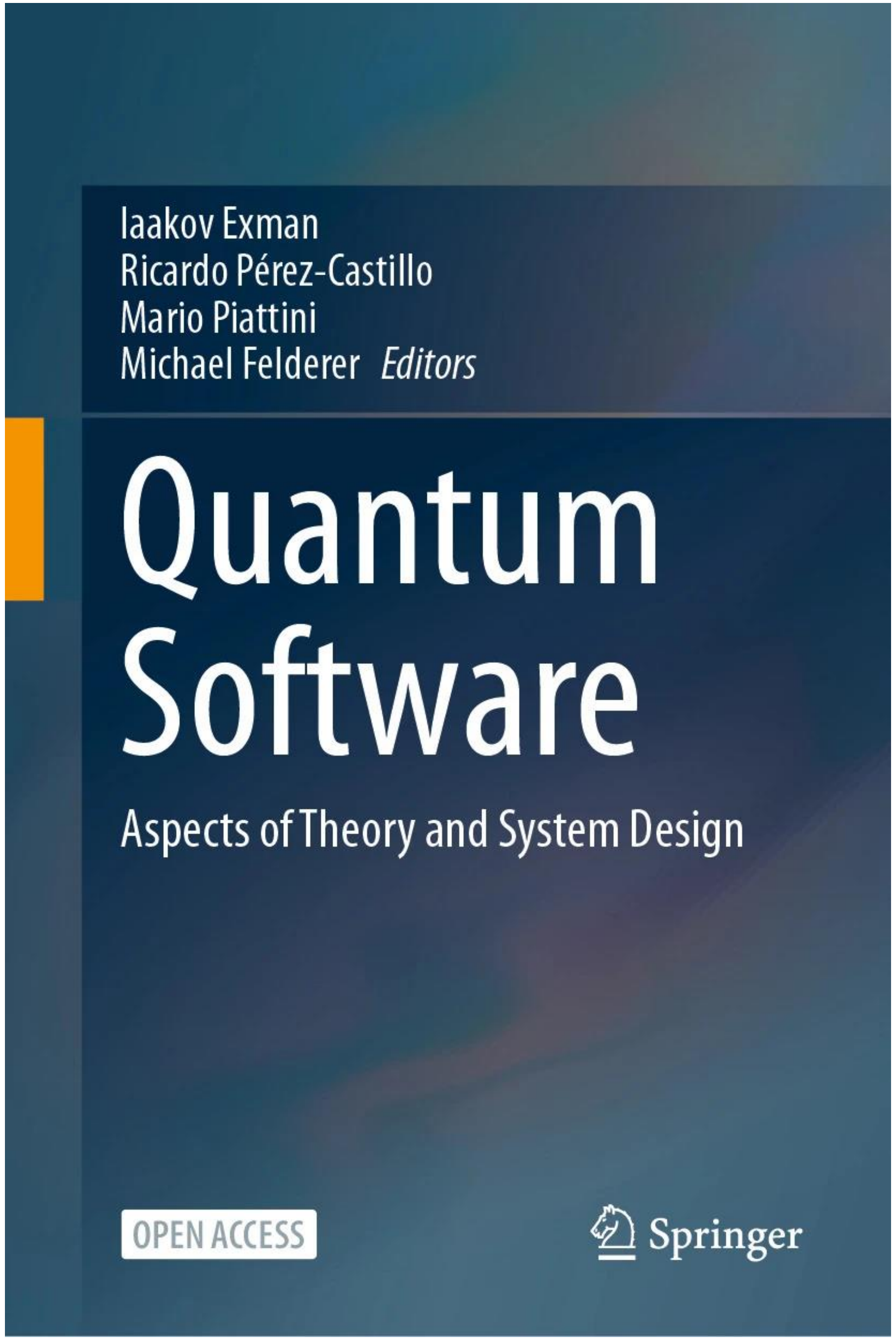
# QUANTUM SOFTWARE ECOSYSTEM DESIGN

Achim Basermann, Michael Epping, Benedikt Fauseweh, Michael Felderer, Elisabeth Lobe, Melven Röhrig-Zöllner, Gary Schmiedinghoff, Peter K. Schuhmacher, Yoshinta Setyawati & Alexander Weinert

DLR Institut für Softwaretechnologie

DLR Quantencomputing Initiative




# Quantum Software Ecosystem Design

**Achim Basermann, Michael Epping, Benedikt Fauseweh, Michael Felderer, Elisabeth Lobe, Melven Röhrig-Zöllner, Gary Schmiedinghoff, Peter K. Schuhmacher, Yoshinta Setyawati, and Alexander Weinert**

**Abstract** The rapid advancements in quantum computing necessitate a scientific and rigorous approach to the construction of a corresponding software ecosystem, a topic underexplored and primed for systematic investigation. This chapter takes an important step in this direction. It presents scientific considerations essential for building a quantum software ecosystem that makes quantum computing available for scientific and industrial problem-solving. Central to this discourse is the concept of hardware–software co-design, which fosters a bidirectional feedback loop from the application layer at the top of the software stack down to the hardware. This approach begins with compilers and low-level software that are specifically designed to align with the unique specifications and constraints of the quantum processor, proceeds with algorithms developed with a clear understanding of underlying hardware and computational model features, and extends to applications that effectively leverage the capabilities to achieve a quantum advantage. We analyze the ecosystem from two critical perspectives: the conceptual view, focusing on theoretical foundations, and the technical infrastructure, addressing practical implementations around real quantum devices necessary for a functional ecosystem. This approach ensures that the focus is toward promising applications with optimized algorithm–circuit synergy, while ensuring a user-friendly design, an effective data management, and an overall orchestration. This chapter thus offers a guide to the essential concepts and practical strategies necessary for developing a scientifically grounded quantum software ecosystem.

**Keywords** Quantum computing · Software ecosystem · Hardware–software co-design · Software engineering

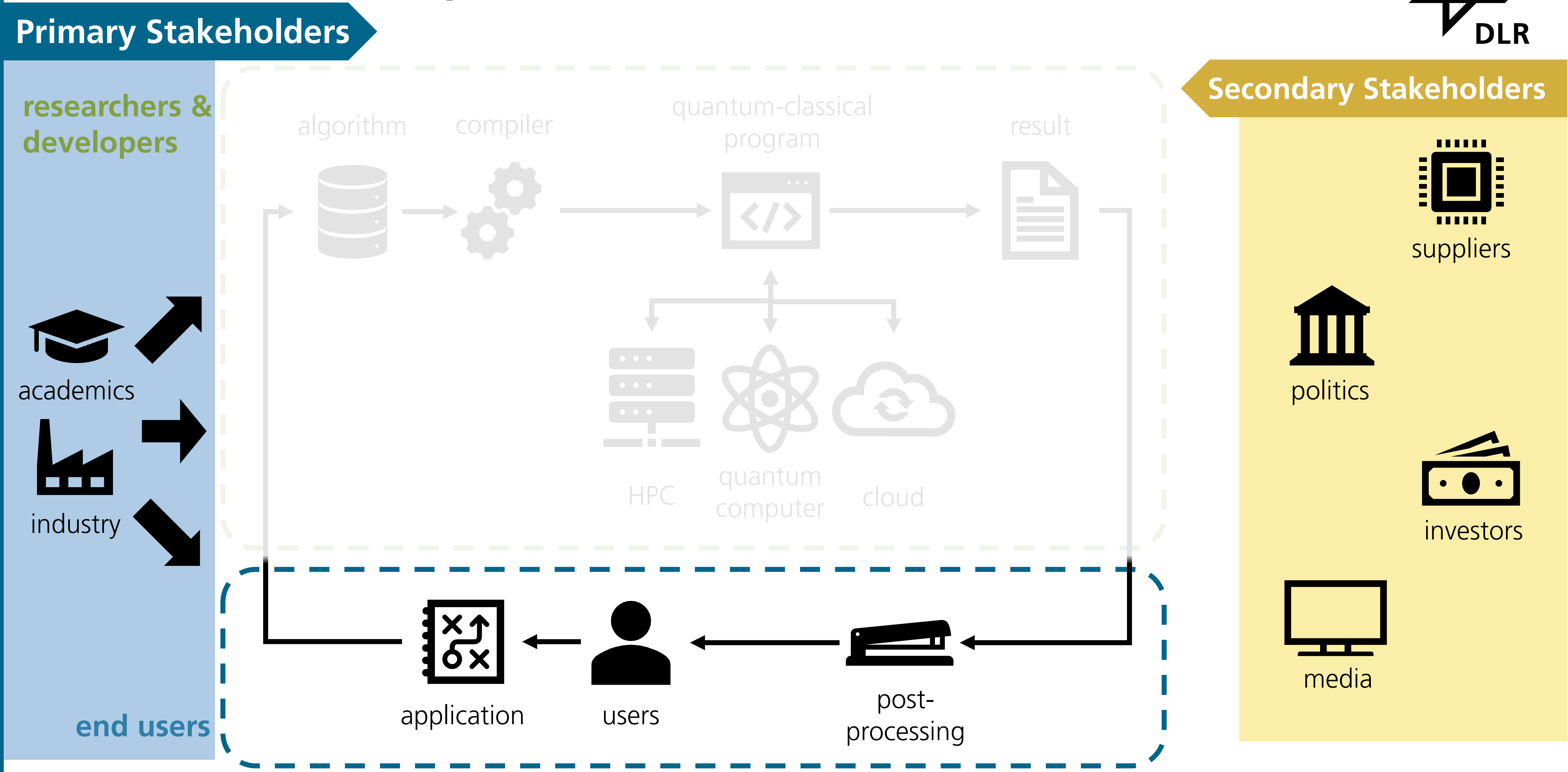
A. Basermann · M. Epping · B. Fauseweh · M. Felderer · E. Lobe (✉) · M. Röhrig-Zöllner · G. Schmiedinghoff · P. K. Schuhmacher · Y. Setyawati · A. Weinert
Institute of Software Technology, German Aerospace Center (DLR), Cologne, Germany
e-mail: elisabeth.lobe@dlr.de


I. Exman et al. (eds.), *Quantum Software*,
https://doi.org/10.1007/978-3-031-64136-7_7

iStock.com

# THE VISION

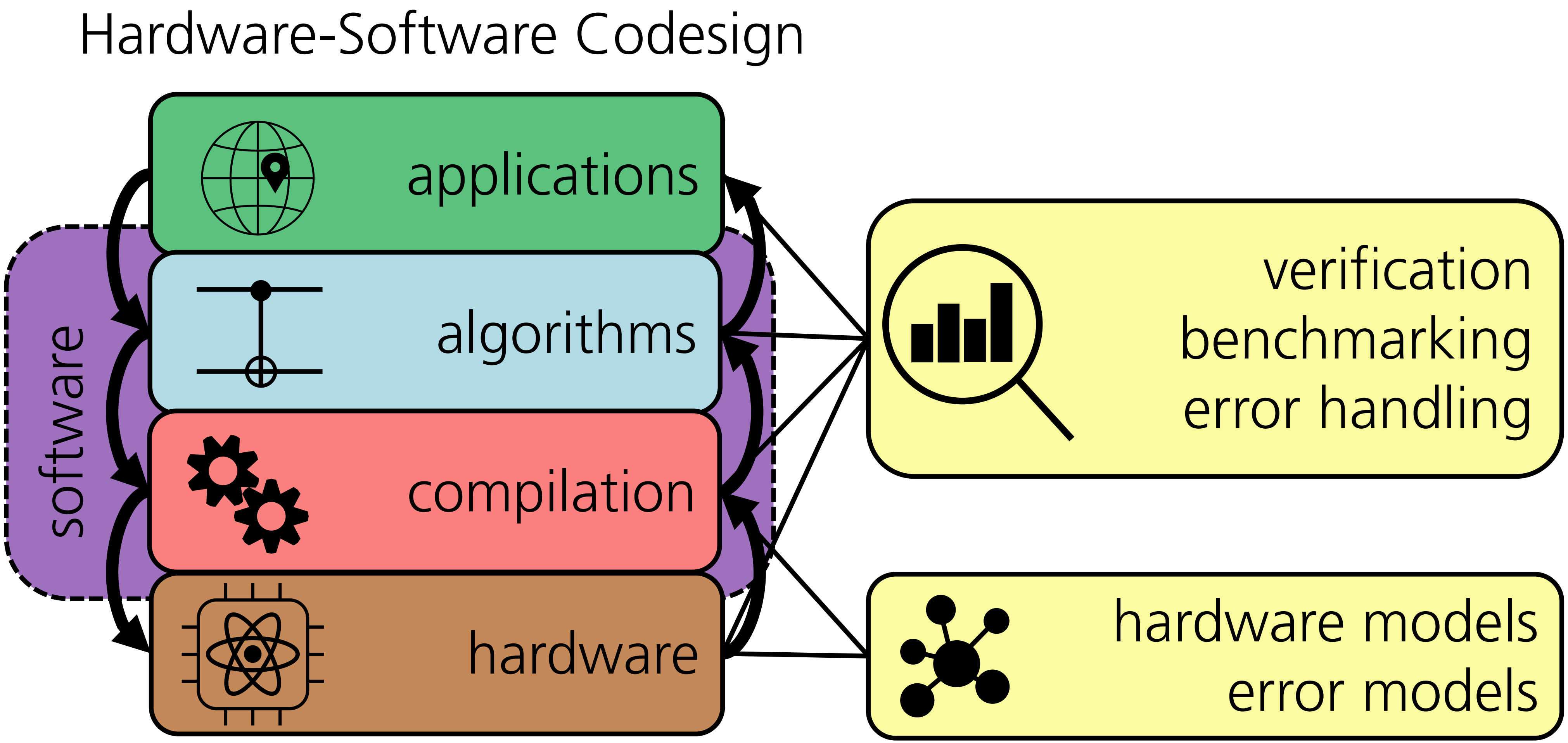
# Intended User Experience



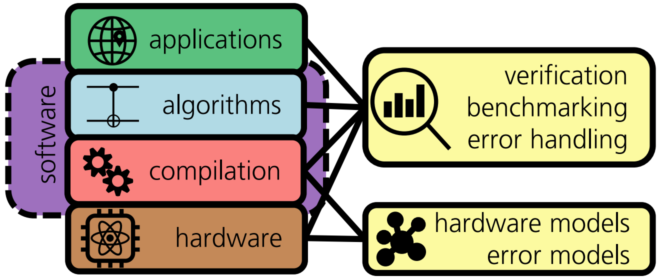
# CONCEPTUAL VIEW

# TECHNICAL VIEW

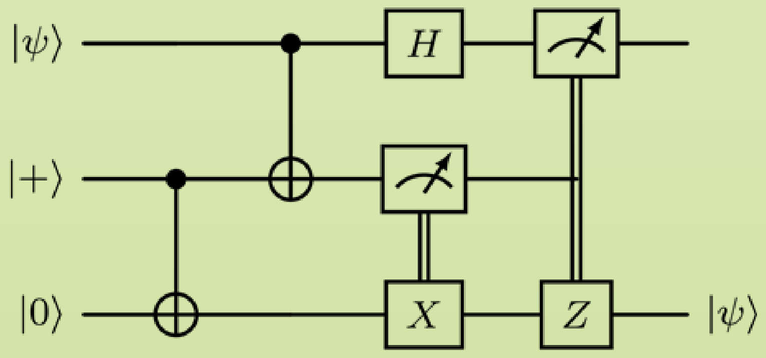
# Conceptual Stack



# Computational Paradigms



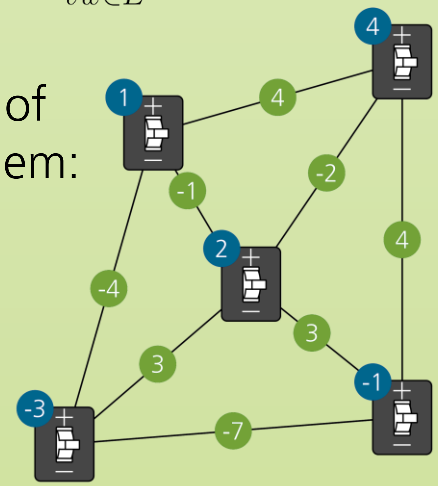

## Gate-Based QC




- manipulation of qubit registers with gates & measurements
- algorithms represented by circuits
- error models: quantum channels
- non-unitary operations possible, e.g. by measurements

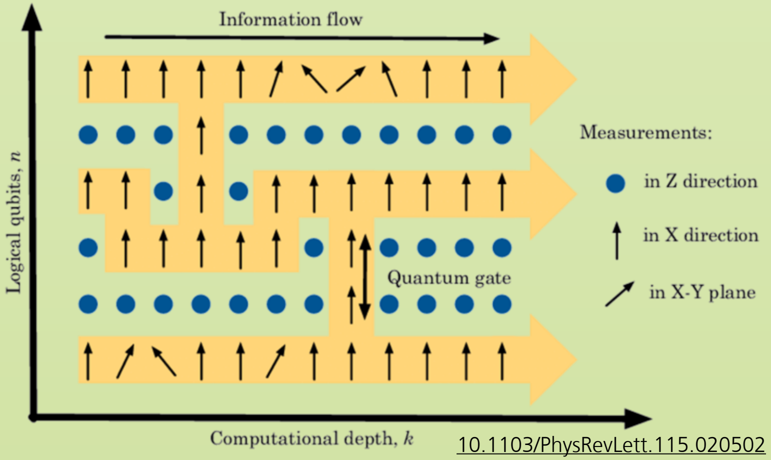
## Adiabatic QC / Q. Annealing

$$\min_{s\in\{-1,1\}^V} \sum_{v\in V} W_v s_v + \sum_{vw\in E} S_{vw} s_v s_w$$




- optimization of specific problem: Ising/QUBO
- single „algorithm“: adiabatic evolution
- only heuristic realization of adiabatic theorem
- sampling from low-energy distribution

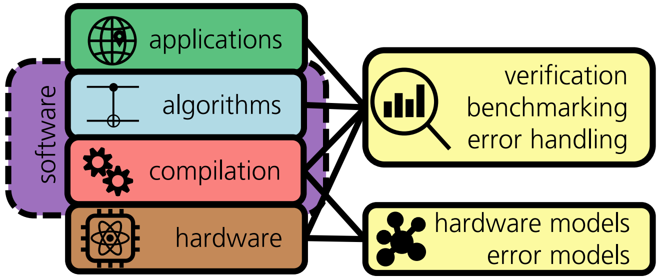
## One-Way QC




10.1103/PhysRevLett.115.020502

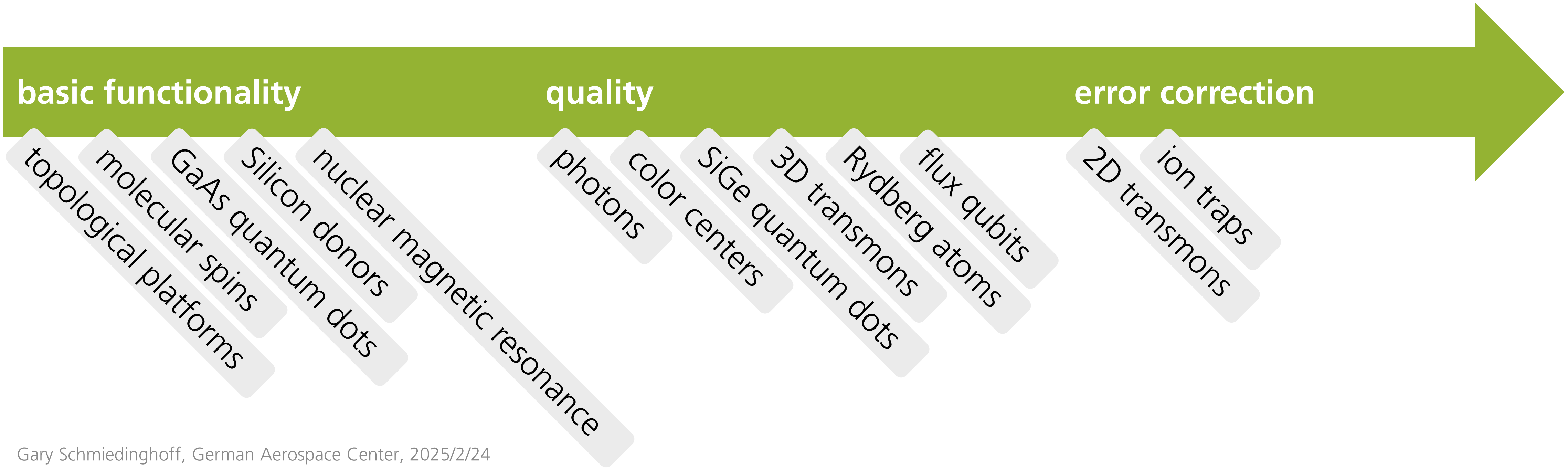
- start with entanglement of large cluster of photons
- operators = measurements and single-qubit rotations
- enormous coherence time but difficult preparation

# Hardware Readiness

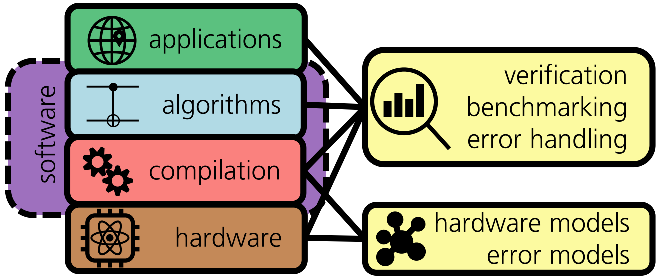



DiVincenzo‘s Criteria:

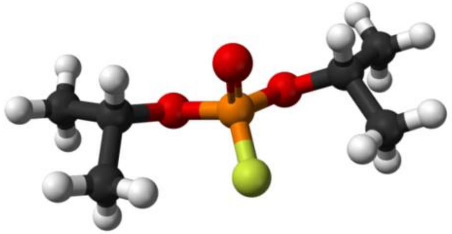
1. scalability with well characterized qubits
2. initialization to simple qubit state
3. long relevant coherence times
4. universal gateset
5. qubit-specific measurement capability



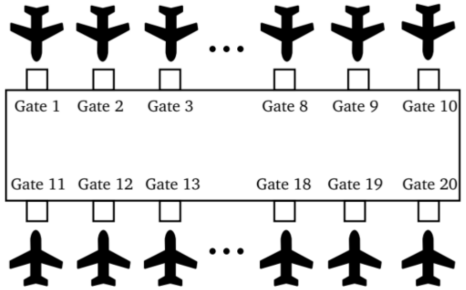
# Promising Applications




## Quantum Simulation



## Combinatorial Optimization




## Quantum-Enhanced ML



## "Classical" Simulation

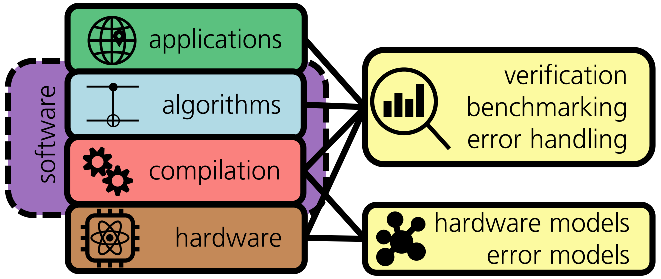


today

time or required error correction

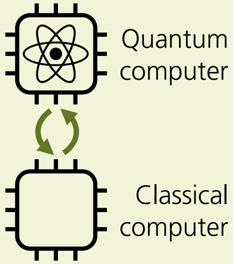
- vibronic structure and dynamics
- atomistic simulation of engineering alloys
- electron transfer in organic photovoltaics
- quantum state compression of hyperspectral data
- uncertainty in the calculation of glacial ice mass balances
- system modelling in solar energy research
- transmission expansion problem
- multi-robotic fibre composite lightweight construction
- loading optimization for autoclave processes

# Algorithms




## (Hybrid) Algorithms for NISQ Devices

- goal: speed-up even for early devices
- often heuristic algorithms




| Variational Quantum Eigensolver | Quantum Approximate Optimization | Quantum Imaginary Time Evolution | |
|---|---|---|---|
| simulating molecules and solid state systems | combinatorial optimization | simulating molecules and solid state systems | … |

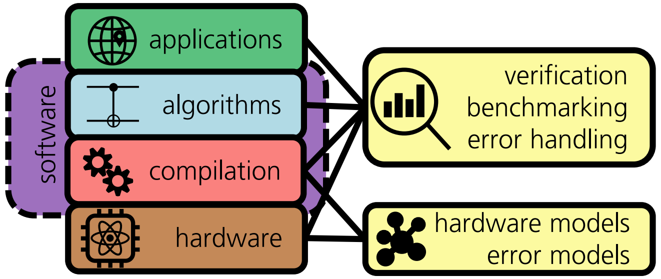
## Powerful Algorithms for Fault-Tolerant Devices

- long-term goal of significant, proven speed-up

| Shor Algorithm | Quantum Fourier Transformation | Grover Algorithm | |
|---|---|---|---|
| prime factorization ➔ encryption solving | signal processing, frequency analysis, … | unsorted data-base search, amplitude am-plification, … | … |

an extensive quantum software library needs to include various algorithms / algorithmic building blocks

# Compiling for Gate-Based QC




**Synthesis**

from unitary matrix to gate sequence

**Transpilation**

to native gateset

**Routing**

to match hardware connectivity

**Optimization**

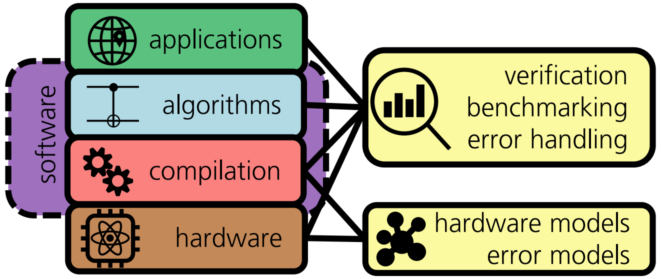
reduce gates in noise-aware way

**Hybrid Compilation**

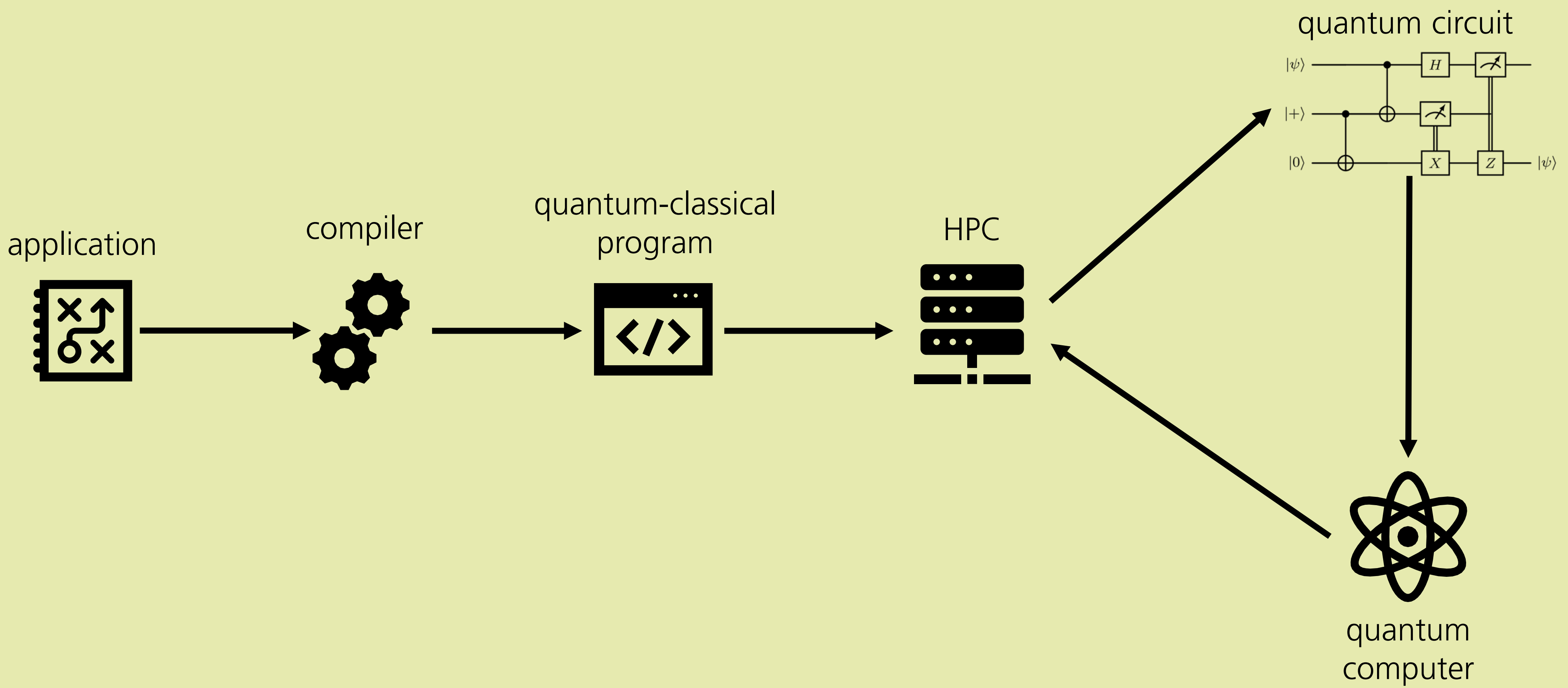
inside/outside coherence time

Produce correct, efficient, hardware-compatible output
for quantum and classical parts

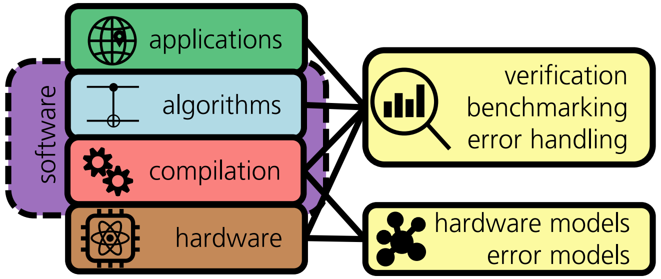
# Compiling for Gate-Based QC




## Hybrid Compilation

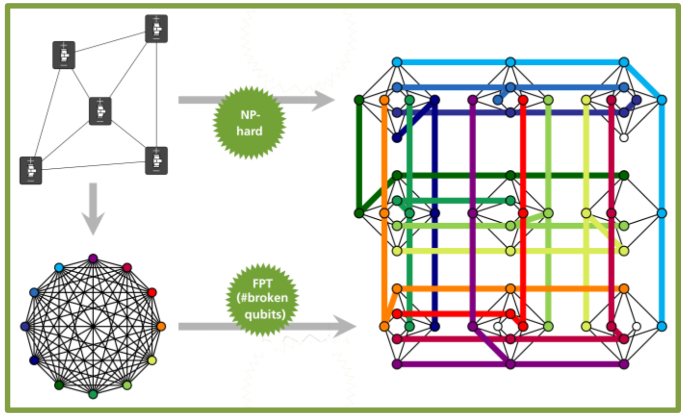


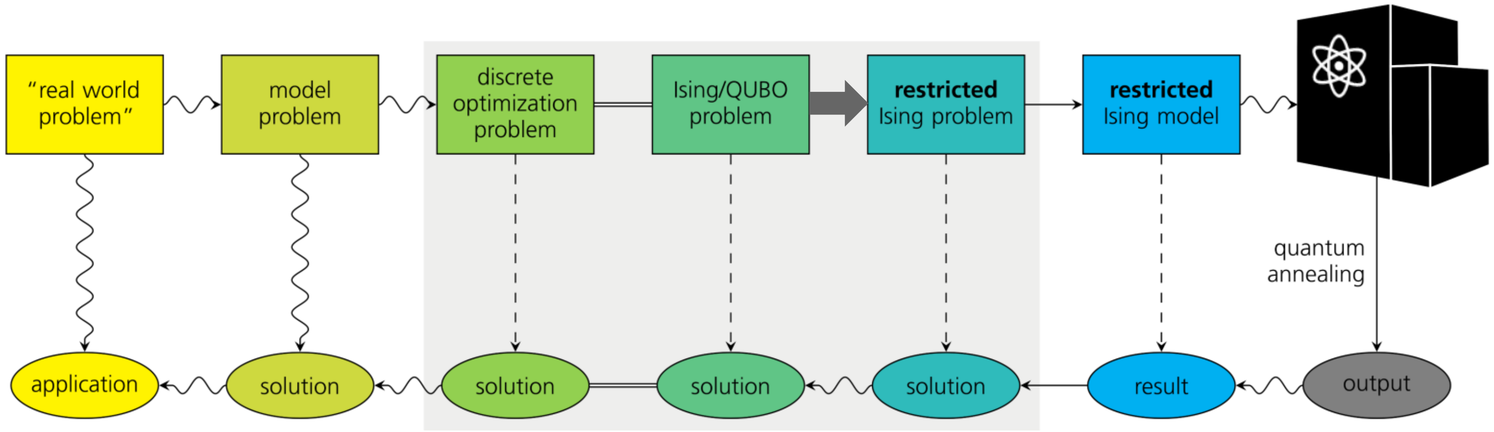
# “Compiling” for Quantum Annealers

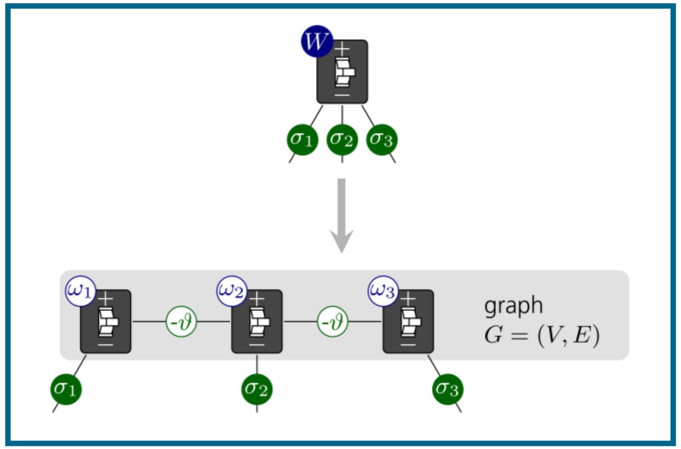



- programming = providing problem-defining parameters
- no general, trivial way of obtaining parameters
- two-step process to restricted Ising problem:

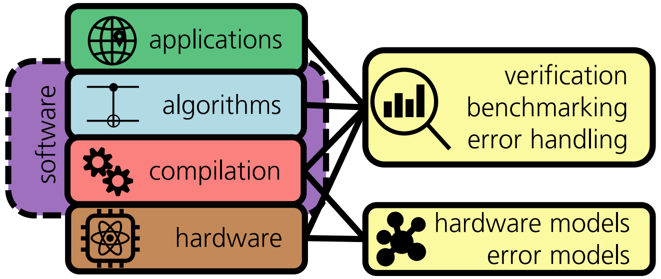
**embedding** and **parameter setting**



# Quantum Software Engineering (QSE)




- **Goal:** develop QSE method even before complex quantum software appears
- **Challenge:** key differences exist between QC software & classical software

## Aspects Include

### Requirement Engineering

- involves domain experts
- hardware sets limits

### Software Design

- **hybrid formulations**
- interfaces & encoding schemes
- **ensure reusability!**

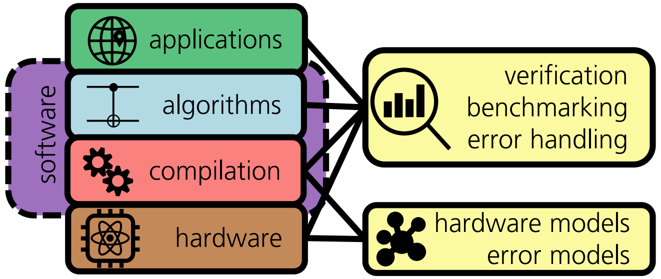
### Models & Representations

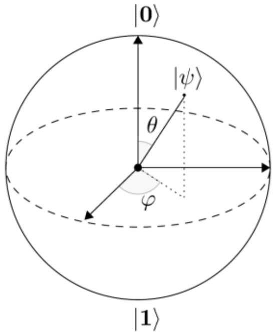
- circuit models
- intermediate representations
- **high-level languages**

### Testing

- new typical errors
- no-cloning theorem
- measurements

# Software-Based Error Handling




Continuous phases & amplitudes inherently fragile!

Application

## Error Correction

- corrects bit- & phase-flips
- uses redundant qubits
- requires noise below threshold

## Error Mitigation

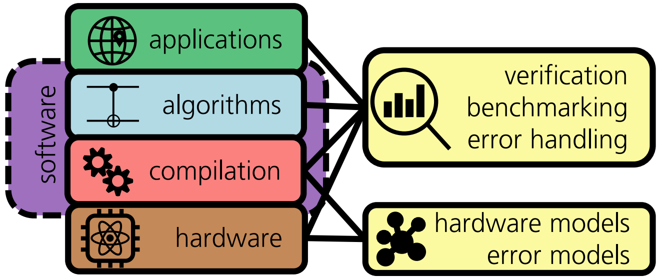
- post-processing of results
- use extra measurements
- examples:
  - zero-noise extrapolation
  - readout error mitigation

## Error Suppression

- improves control
- examples:
  - dynamical decoupling
  - spin-echo pulses

Hardware

# Quantum Software Verification




## Does our Software Fulfill its Requirements?

**Challenge:**
stochastic nature + noise

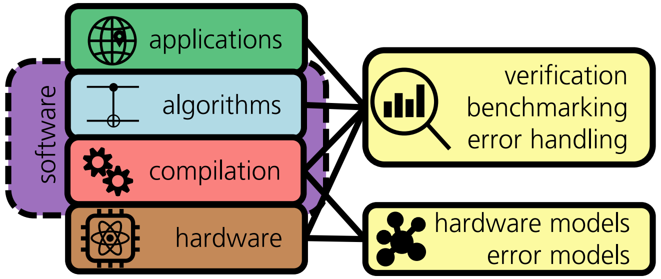
**Goal:**
high-quality solution
(high probability close to the desired result)

## Required Ingredients

| Theoretical Proof | Practical Validation | Working Toolchain |
|---|---|---|
| *quantum algorithm satisfies pre- and postconditions* | *code implements desired algorithm* (quantum + classical part) | *code translated correctly to executable quantum circuit* (+ correctly working hardware) |

**Research Question:**
If quantum computers outperform classical computers,
how can we ensure correctness?

# Benchmarking




DLR

## Goal: Quantify the Performance of Soft- and Hardware

### Current Limitations

- tailored to specific device
- only cover single aspects
- not “how fast” but “how good”

**Benchmarks only give information about current devices!**

### Typical Metrics

- **hardware metrics:**
  - # qubits
  - connectivity
  - gateset
- **quality metrics:**
  - coherence times
  - gate/circuit fidelities
  - quantum volume

### Typical Methods

- state and process tomography
- randomized benchmarking (randomly insert gates that allow efficient classical simulation)

### Desired Future Insights

- distinct standard suites to assess:
  - performance and correctness (comparison between quantum HW & SW)
  - quantum advantage (fastest QC vs. fastest classical)
  - near-term practicability (cost-to-solution of application)

# TECHNICAL VIEW

iStock.com

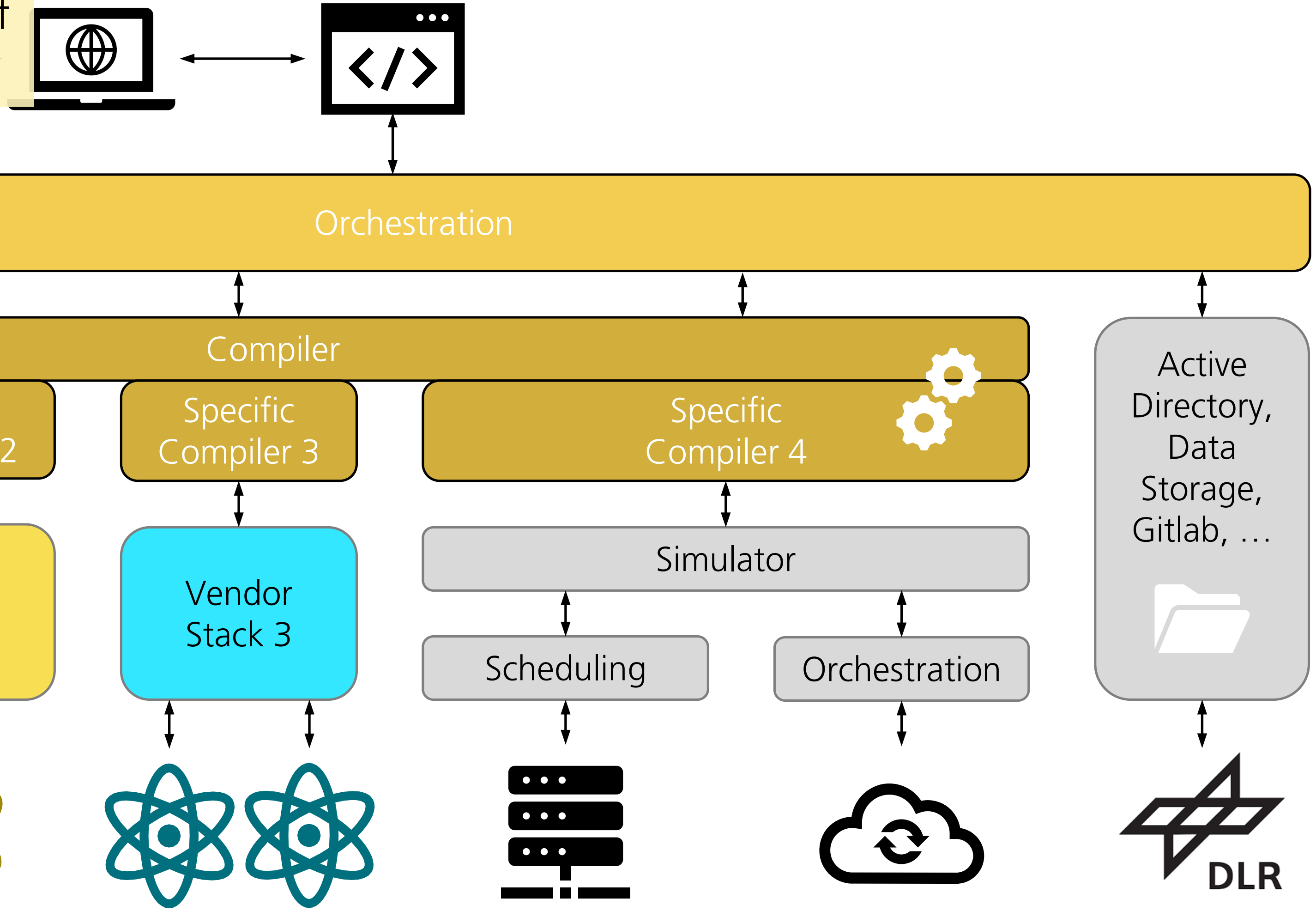
# Technical View

## Intuitive User Interface

- support rapid iteration
- must allow requirement definition of several composable services

## Orchestration and Data Management

- orchestrate classical & quantum execution
- schedule use of QC, HPC, cloud resources
- store & manage data and requirements



# Industry Partners

| Quantum Software Developers | Hardware Producer for Ion Trap QC | Industrial End Users from Material Sciences |
| --- | --- | --- |
| d-fine | eleQtron | HQS Quantum Simulations |
| planqc | parityqc | IQM |
| | NXP | Multiverse Computing |
| | | Single Quantum |

*Thank you for listening*

# Imprint

| | |
|---|---|
| Topic: | **Quantum Software Ecosystem Design**<br>Overview of the book chapter |
| Date: | 2025-02-24 |
| Author: | Gary Schmiedinghoff |
| Institute: | German Aerospace Center (DLR) – Institute of Software Technology |
| Image sources: | All images “DLR (CC BY-NC-ND 3.0)” unless otherwise stated |