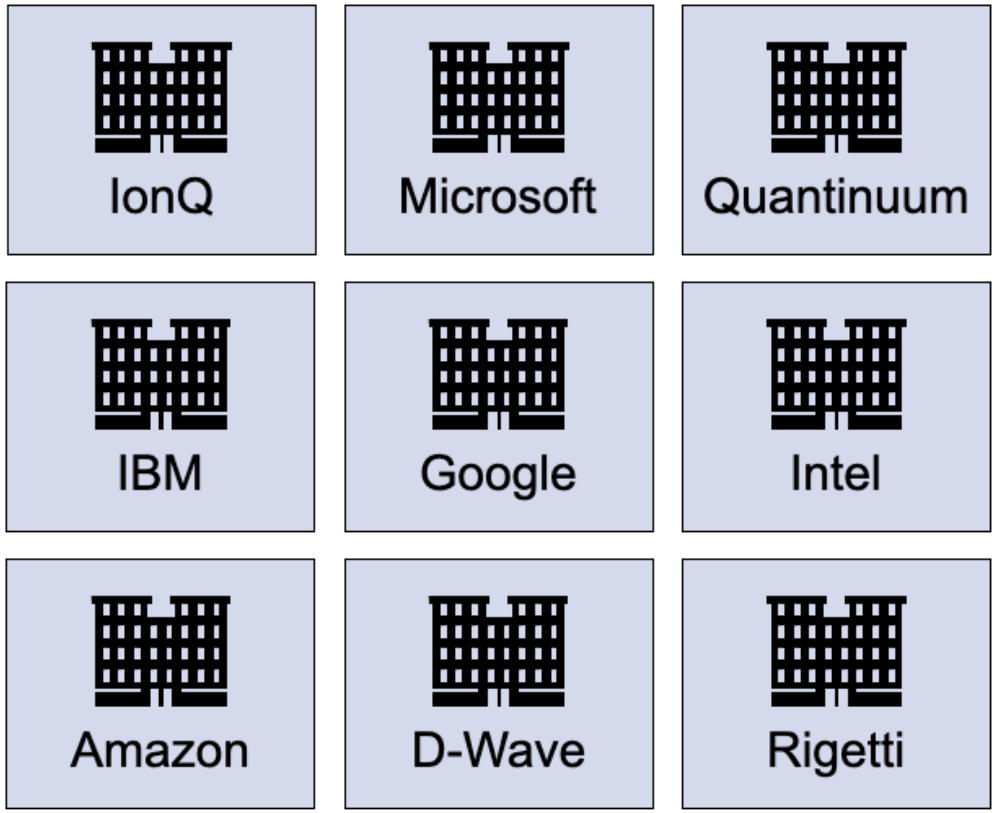
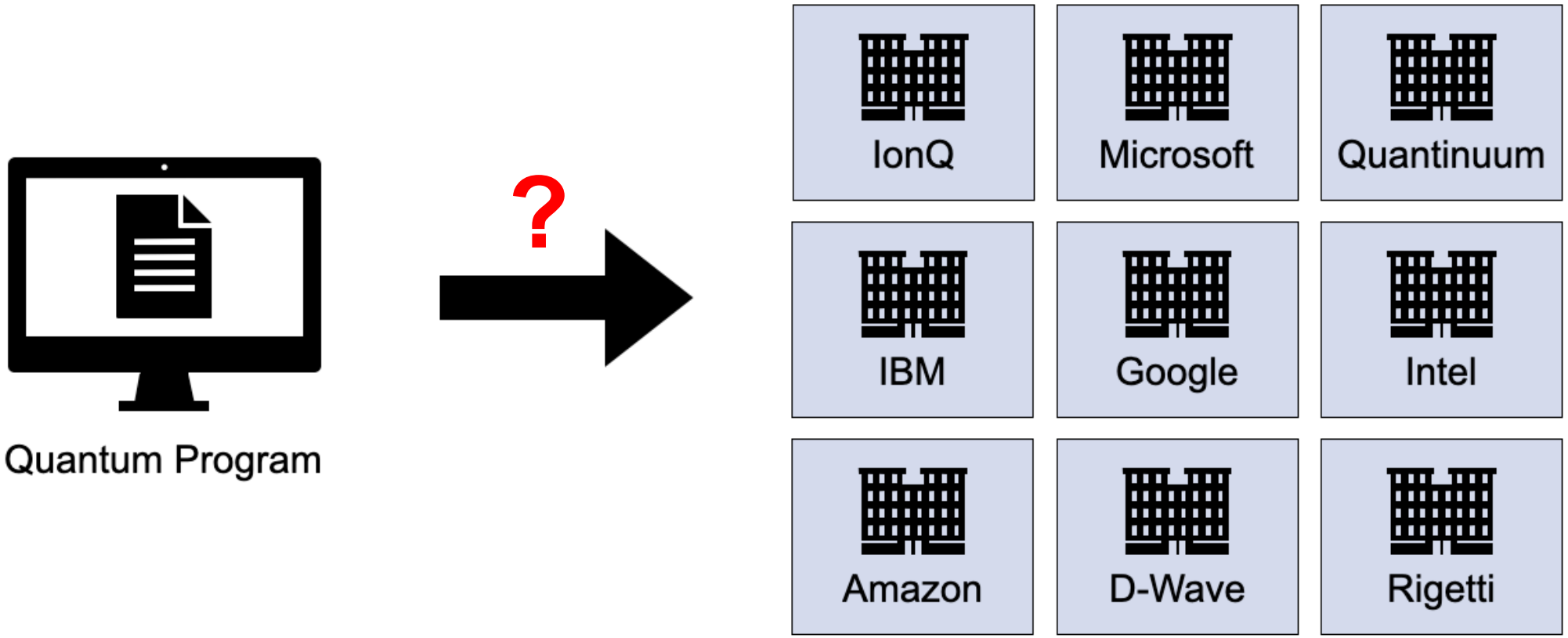
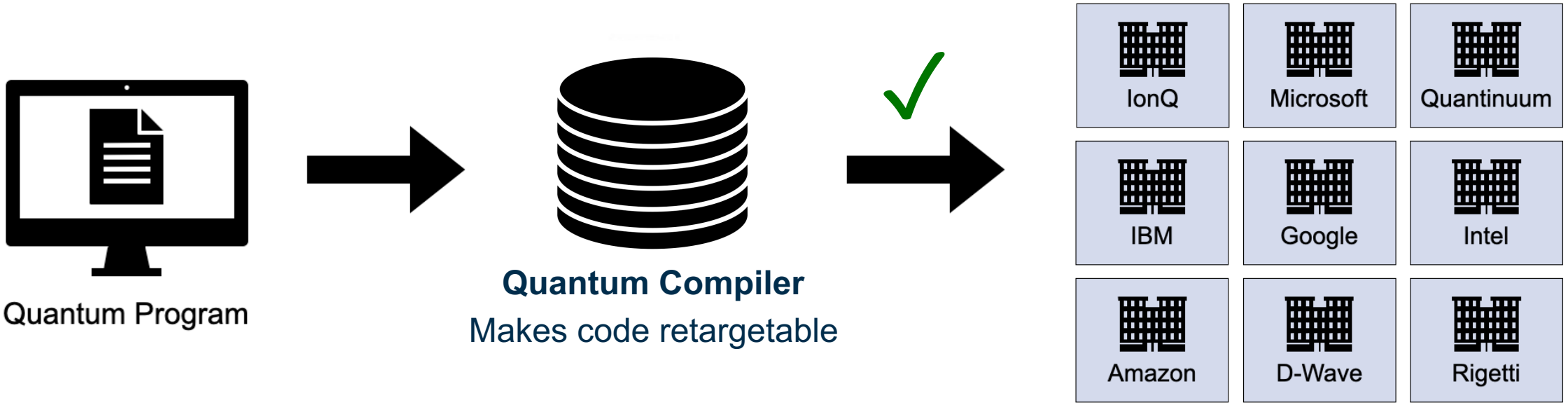
Investigating Retargetability Claims for Quantum Compilers

Luke Southall, Joshua Ammermann, Rinor Kelmendi,
Domenik Eichhorn, Ina Schaefer

Domenik Eichhorn, TVA, KASTEL
23.02.2026


IonQ
Microsoft
Quantinuum
IBM
Google
Intel
Amazon
D-Wave
Rigetti


Quantum Program
?
IonQ
Microsoft
Quantinuum
IBM
Google
Intel
Amazon
D-Wave
Rigetti


Quantum Program
Quantum Compiler
Makes code retargetable
IonQ
Microsoft
Quantinuum
IBM
Google
Intel
Amazon
D-Wave
Rigetti

Contributions and Research Questions

Retargetability?

Contributions and Research Questions

Retargetability?

Develop methodology to measure retargetability of a quantum compiler

Contributions and Research Questions

Retargetability?

Develop methodology to measure retargetability of a quantum compiler

Apply this methodology to empirically assess three popular quantum compilers regarding retargetability: Tket, Qiskit, and ProjectQ

Contributions and Research Questions

Develop methodology to measure retargetability of a quantum compiler

Apply this methodology to empirically assess three popular quantum compilers regarding retargetability: Tket, Qiskit, and ProjectQ

RQ1: What are important aspects of retargetability, and how can we measure them?

RQ2: How retargetable are existing quantum compilers according to our methodology?

Contributions and Research Questions

Develop methodology to measure retargetability of a quantum compiler

Apply this methodology to empirically assess three popular quantum compilers regarding retargetability: Tket, Qiskit, and ProjectQ

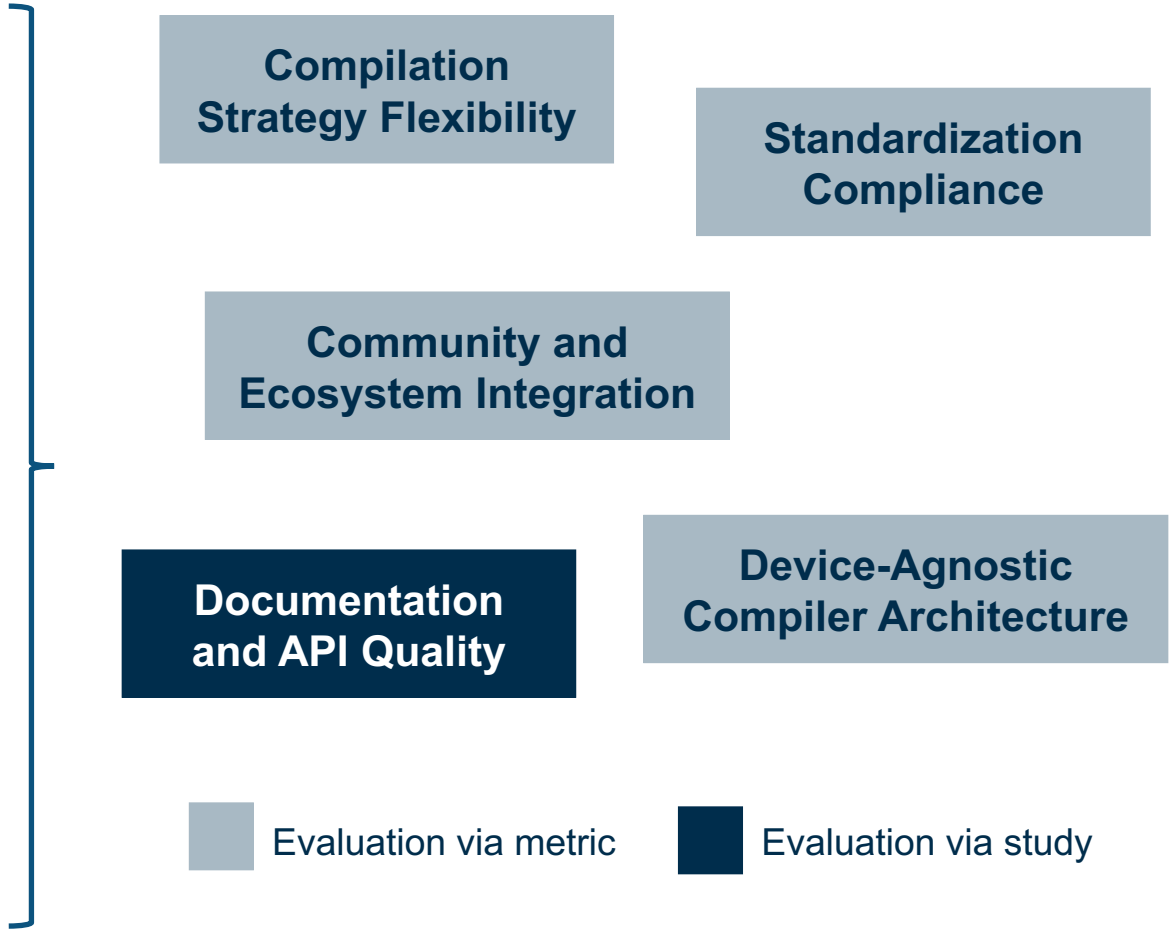
RQ1: What are important aspects of retargetability, and how can we measure them?

RQ2: How retargetable are existing quantum compilers according to our methodology?

Measuring Retargetability

Important Aspects of Retargetability

- **For classical compilers:**
  - Modular and device-agnostic architectures
  - Flexible optimization pipelines
- **Quantum space:**
  - Lack of standardization hindering better retargetability
- **Generally:**
  - Comprehensive API and documentation necessary to achieve **practical retargetability**



Metric-based Assessment of Retargetability

Compilation Strategy Flexibility | Standardization Compliance | Community and Ecosystem Integration | Device-Agnostic Compiler Architecture

- **For better retargetability:** fine-grained control over compilation processes while remaining hardware independent

- **We rate:**
  - Level of control for user over compilation passes
  - Level of pre-built compilation passes available for selection and configuration
  - Level of options for specifying hardware-specific constraints
  - Level of available preset compilation strategies
  - Level of support for creating and integrating custom compilation passes

- Each question scored between 1 and 5, where **higher scores are better**

Metric-based Assessment of Retargetability

Compilation Strategy Flexibility	**Standardization Compliance**	Community and Ecosystem Integration	Device-Agnostic Compiler Architecture

- **For better retargetability:** more uniform standards or adoption of (quasi-)standards to improve interoperability

- **We rate:**
  - Level of support for OpenQASM
  - Level of support for QIR
  - Level of support for additional interchangeable formats
  - Level of active involvement in standardization efforts
  - Level of overall interoperability with other quantum software tools

- Each question scored between 1 and 5, where **higher scores are better**

Metric-based Assessment of Retargetability

Compilation Strategy Flexibility	Standardization Compliance	**Community and Ecosystem Integration**	Device-Agnostic Compiler Architecture

- **For better retargetability:** active community and good integration with hardware providers, due to fast-moving research field

- **We rate:**
  - Level of active integration with providers
  - Level of growth in supported integrations
  - Ease of use of the provided extension system
  - Effectiveness of backend distribution
  - Level of community engagement

- Each question scored between 1 and 5, where **higher scores are better**

Metric-based Assessment of Retargetability

Compilation Strategy Flexibility	Standardization Compliance	Community and Ecosystem Integration	**Device-Agnostic Compiler Architecture**

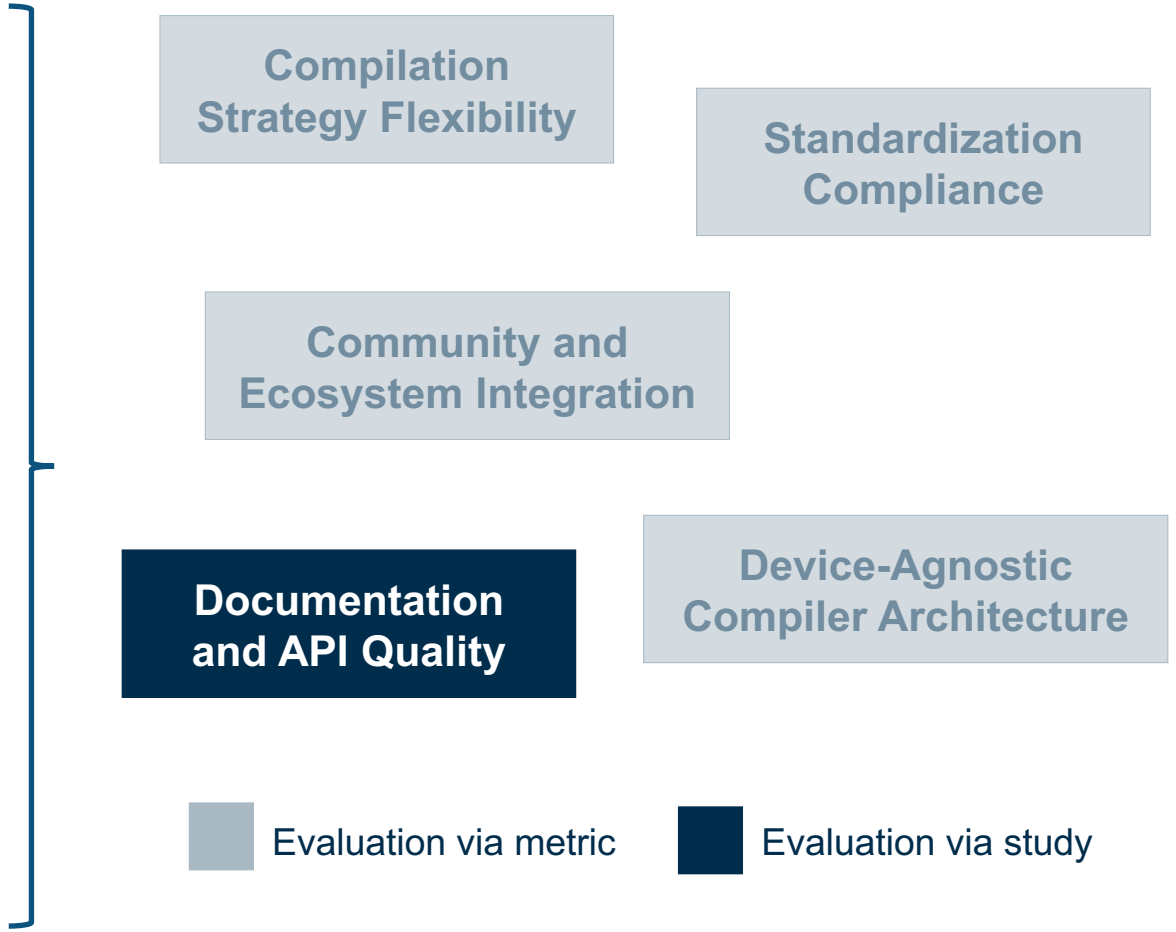
- **For better retargetability:** no coupling of compilers to specific manufacturer architectures, implementation of device-agnosticism

- **We rate:**
  - Level of modular design
  - Level of hardware-agnostic design
  - Level of adaptability to different quantum gate sets
  - Level of priority of hardware agnosticism in the design
  - Score for codebase assigned by Sonar Qube evaluation (security, reliability, maintainability)

- Each question scored between 1 and 5, where **higher scores are better**

Important Aspects of Retargetability

- **For classical compilers:**
  - Modular and device-agnostic architectures
  - Flexible optimization pipelines
- **Quantum space:**
  - Lack of standardization hindering better retargetability
- **Generally:**
  - Comprehensive API and documentation necessary to achieve **practical retargetability**



User Study for Documentation and API Quality

Documentation and API Quality

- **Aim of the study:** assess practical retargetability from a developer's perspective

- **Study design:**
  - Developers solve implementation task: create new backend for quantum compiler (here: simulator)
  - After implementation, developers answer questionnaire [So25]
  - For each compiler, possible hurdles identified beforehand
  - Hurdles used to create hints, that participants can use for the implementation (Designed to simulate community forums or similar, for higher internal validity)

- **Study protocol:**
  - Standardized instructions for each compiler (task description, relevant documentation, basic setup)
  - Questionnaire designed using best practices from [DSC14]

[So25] Southall, L. et al.: Methodology for Retargetability Assessment of Quantum Compilers, 2025, https://doi.org/10.5281/zenodo.17780085.
[DSC14] Dillman, D. A.; Smyth, J. D.; Christian, L. M.: Internet, Phone, Mail, and Mixed-Mode Surveys: The Tailored Design Method. Wiley Publishing, 2014.

User Study for Documentation and API Quality

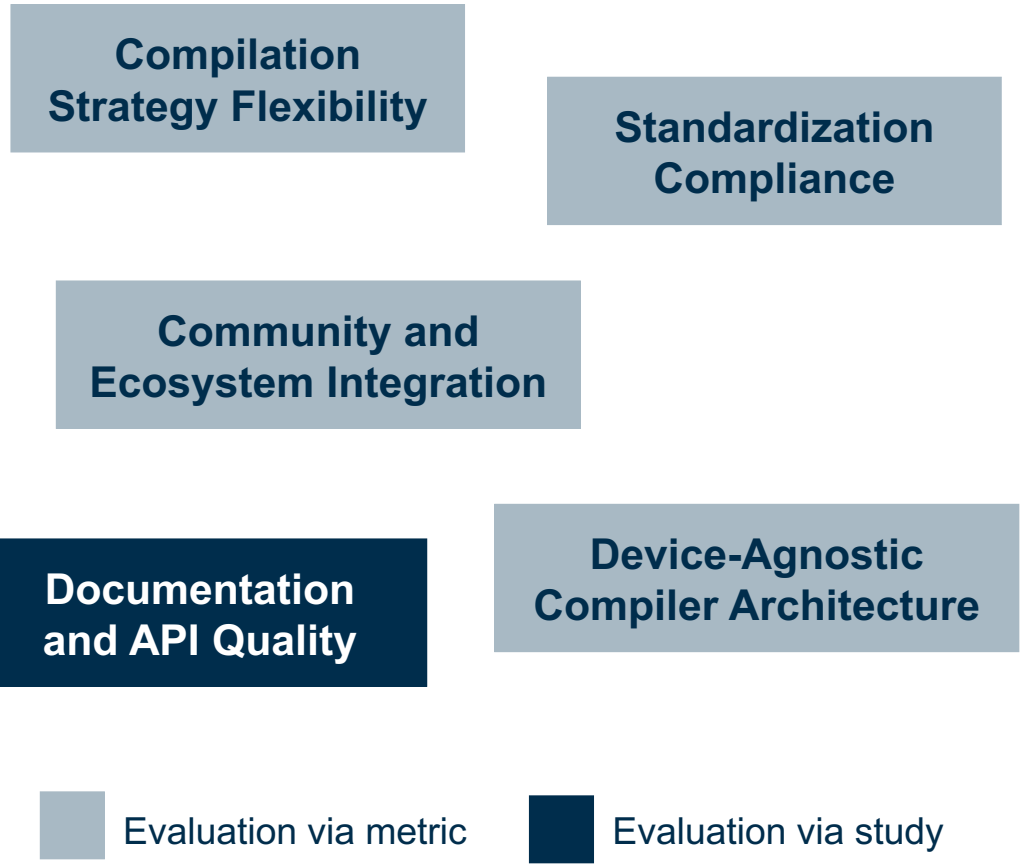
Documentation and API Quality

- **Evaluated sub-dimensions:**
  - Developers prior experience (Q1-Q3)
  - Documentation quality and clarity (Q4-Q8)
  - API clarity (Q9)
  - Overall implementation difficulty (Q10)
  - Hint utility (Q11)

- Evaluation on five-point Likert-scale, where higher scores are better

Overall Evaluation of Retargetability

- **Combination** of metric-based retargetability assessment and study-based retargetability assessment
- This results in **overall retargetability score**:
  - $s_{total} = \sum_{i=1}^{5} w_i \cdot s_i$ where
    $s_i \, \epsilon \, [1, 5]$ is category score and
    $w_i \, \epsilon \, [0, 1]$ is category weight with $\sum_{i=1}^{5} w_i = 1$.
  - In our case: $w_{1..5} = 0.2$
- In our case: **arithmetic mean**, but could be adapted



Contributions and Research Questions

Develop methodology to measure retargetability of a quantum compiler

Apply this methodology to empirically assess three popular quantum compilers regarding retargetability: Tket, Qiskit, and ProjectQ

RQ1: What are important aspects of retargetability, and how can we measure them?

RQ2: How retargetable are existing quantum compilers according to our methodology?

Contributions and Research Questions

Develop methodology to measure retargetability of a quantum compiler

Apply this methodology to empirically assess three popular quantum compilers regarding retargetability: Tket, Qiskit, and ProjectQ

RQ1: What are important aspects of retargetability, and how can we measure them?

RQ2: How retargetable are existing quantum compilers according to our methodology?

Retargetability Assessment: Tket, Qiskit and ProjectQ

Retargetability Assessment

- Apply established **methodology from RQ1**

- **Compiler selection:**
  - **Tket:** explicit claims of retargetability
  - **Qiskit:** popularity, available guide on implementing custom backends
  - **ProjectQ:** though no explicit retargetability claim, hardware-agnosticism and custom backend integration

1. Metric-based retargetability evaluation
2. User-study, implementing simulator backend

Tket

Qiskit

ProjectQ

Retargetability Assessment: Metric-based approach
Compilation Strategy Flexibility

Question	Tket	Qiskit	ProjectQ
How much or how little control does a user have over compilation passes (selection, ordering, customization)?	5	5	3
How extensive or limited are the pre-built compilation passes available for selection and configuration?	5	5	3
How comprehensive or limited are the options for specifying and customizing hardware-specific constraints (e.g., qubit connectivity, gate fidelities)?	5	5	2
How extensive or limited are the preset compilation strategies available (e.g., noise mitigation, depth reduction)?	5	5	2
How robust or limited is the support for creating and integrating custom compilation passes?	5	5	5
Average	5	5	3

Main Takeaway:
TKET and **Qiskit** offer consistently **high and comprehensive compilation strategy flexibility**,
while **ProjectQ** provides more **limited flexibility** overall, particularly in hardware-specific customization and preset strategies.

Retargetability Assessment: Metric-based approach
Standardization Compliance

Question	Tket	Qiskit	ProjectQ
How complete or incomplete is the compiler's support for QIR?	4	2	1
How complete or incomplete is the compiler's support for OpenQASM (import and export)?	3	4	1
How extensive or limited is the compiler's support for additional quantum-circuit description formats (e.g. Quil)?	4	2	1
How active or inactive is the compiler's involvement in quantum-computing standardization efforts?	5	5	1
How well or poorly does the compiler interoperate with other quantum frameworks (e.g., Cirq)?	5	4	2
Average	4.2	3.4	1.2

Main Takeaway:
TKET leads in interoperability and standards engagement, **Qiskit** shows solid but more uneven format support, and **ProjectQ** offers significantly more limited ecosystem integration overall.

Retargetability Assessment: Metric-based approach
Device-Agnostic Compiler Architecture

Question	Tket	Qiskit	ProjectQ
How modular or monolithic is the compiler's overall architecture?	5	5	5
How hardware-agnostic or hardware-specific are the compiler's components?	5	5	5
How adaptable or limited is the compiler to diverse quantum gate sets?	5	5	5
How high or low is the priority of hardware-agnostic design within the compiler?	5	5	5
What score did the compiler's codebase receive in the Sonar Qube evaluation?	4	2.3	2.67
Average	4.8	4.46	4.53

Main Takeaway:
All three compilers offer highly device-agnostic architectures.
The SonarQube evaluation produced positive results for Qiskit and revealed several issues for both Qiskit and ProjectQ.

Retargetability Assessment: Metric-based approach
Community and Ecosystem Integration

Question	Tket	Qiskit	ProjectQ
How well or poorly integrated is the compiler with major and minor quantum-hardware providers?	5	5	4
How strong or weak has the growth been in the number of supported backends over time?	5	5	2
How extensive or limited are the compiler's extension capabilities (e.g., plugin system)?	5	5	4
How easy or difficult is it to distribute a custom backend with the compiler?	5	5	2
How active or inactive is the community surrounding the compiler (e.g., commits, contributors, discussions)?	5	5	2
Average	5	5	2.8

Main Takeaway:
Qiskit and **Tket** are well-established within the community and boast a wealth of ecosystem integrations.
ProjectQ falls short in terms of the number of supported backend, and its community is comparatively inactive.

Retargetability Assessment: User-study based approach

- **Participants:** six post- and undergraduate students of computer science
  - Basic knowledge of quantum software engineering
  - Using own device for the study, used either MacOS or Windows
  - All participants invested about two working days into the implementation tasks

- **Instructions:** Provided for each compiler, containing basic instructions

- **Task:**
  - Implement backend for Stim quantum simulator for each compiler
  - No referral to the internet or similar, apart from links provided on the task sheets
  - After each implementation, fill out questionnaire for this compiler

Retargetability Assessment: User-study based approach

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4						
	P5	2	2	1	4	4	3						
	P6	4	3	1	4	4	3						
		3.67	3.5	1.83	4.33	4	3.83						
ProjectQ	**P1**	3	1	1	2	1	2						
	P2	5	1	1	1	1	1						
	P3	3	1	1	1	1	1						
	P4	5	1	1	1	1	2						
	P5	2	1	1	3	3	2						
	P6	4	1	1	1	1	1						
		3.67	1	1	1.5	1.33	1.5						

Documentation and API Quality

- **Evaluated sub-dimensions:**
  - Developers prior experience (Q1-Q3)
  - Documentation quality and clarity (Q4-Q8)
  - API clarity (Q9)
  - Overall implementation difficulty (Q10)
  - Hint utility (Q11)
- Evaluation on five-point Likert-scale, where higher scores are better

Retargetability Assessment: User-study based approach

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4	4	5	3	4	2	
	P5	2	2	1	4	4	3	4	2	4	3	1	
	P6	4	3	1	4	4	3	3	3	4	3	1	
		3.67	3.5	1.83	4.33	4	3.83	3.83	4	3.5	3.83	3.17	**3.81**
ProjectQ	**P1**	3	1	1	2	1	2	2	2	2	2	5	
	P2	5	1	1	1	1	1	2	3	2	1	5	
	P3	3	1	1	1	1	1	1	1	1	1	1	
	P4	5	1	1	1	1	2	2	4	2	1	1	
	P5	2	1	1	3	3	2	4	1	2	2	2	
	P6	4	1	1	1	1	1	1	3	2	1	5	
		3.67	1	1	1.5	1.33	1.5	2	2.33	1.83	1.33	3.17	**1.88**

- **Q1:** Prior experience in quantum software engineering was medium to high
- **Q2:** Prior experience was rated highest for Qiskit and lowest for ProjectQ, although most of this experience not with custom backends
- **Q3:** Prior backend implementation experience generally low, possibly influenced by Tket experience for Qiskit

Retargetability Assessment: User-study based approach

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4	4	5	3	4	2	
	P5	2	2	1	4	4	3	4	2	4	3	1	
	P6	4	3	1	4	4	3	3	3	4	3	1	
		3.67	3.5	1.83	4.33	4	3.83	3.83	4	3.5	3.83	3.17	**3.81**
ProjectQ	**P1**	3	1	1	2	1	2	2	2	2	2	5	
	P2	5	1	1	1	1	1	2	3	2	1	5	
	P3	3	1	1	1	1	1	1	1	1	1	1	
	P4	5	1	1	1	1	2	2	4	2	1	1	
	P5	2	1	1	3	3	2	4	1	2	2	2	
	P6	4	1	1	1	1	1	1	3	2	1	5	
		3.67	1	1	1.5	1.33	1.5	2	2.33	1.83	1.33	3.17	**1.88**

- Over all questions for documentation quality and clarity, Tket performed best
- Qiskit follows up closely behind Tket in these questions
- ProjectQ performs worst among the evaluated compilers for these questions

Retargetability Assessment: User-study based approach

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4	4	5	3	4	2	
	P5	2	2	1	4	4	3	4	2	4	3	1	
	P6	4	3	1	4	4	3	3	3	4	3	1	
		3.67	3.5	1.83	4.33	4	3.83	3.83	4	3.5	3.83	3.17	**3.81**
ProjectQ	**P1**	3	1	1	2	1	2	2	2	2	2	5	
	P2	5	1	1	1	1	1	2	3	2	1	5	
	P3	3	1	1	1	1	1	1	1	1	1	1	
	P4	5	1	1	1	1	2	2	4	2	1	1	
	P5	2	1	1	3	3	2	4	1	2	2	2	
	P6	4	1	1	1	1	1	1	3	2	1	5	
		3.67	1	1	1.5	1.33	1.5	2	2.33	1.83	1.33	3.17	**1.88**

- Tket was rated to have the best API clarity, followed by Qiskit.
- ProjectQ was rated to have a more unclear API

Retargetability Assessment: User-study based approach

		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4	4	5	3	4	2	
	P5	2	2	1	4	4	3	4	2	4	3	1	
	P6	4	3	1	4	4	3	3	3	4	3	1	
		3.67	3.5	1.83	4.33	4	3.83	3.83	4	3.5	3.83	3.17	**3.81**
ProjectQ	**P1**	3	1	1	2	1	2	2	2	2	2	5	
	P2	5	1	1	1	1	1	2	3	2	1	5	
	P3	3	1	1	1	1	1	1	1	1	1	1	
	P4	5	1	1	1	1	2	2	4	2	1	1	
	P5	2	1	1	3	3	2	4	1	2	2	2	
	P6	4	1	1	1	1	1	1	3	2	1	5	
		3.67	1	1	1.5	1.33	1.5	2	2.33	1.83	1.33	3.17	**1.88**

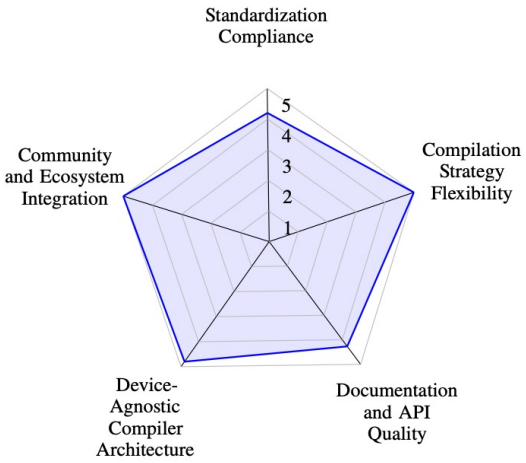
- Qiskit was rated the least difficult to implement, followed by Tket
- ProjectQ was rated as the hardest to implement by a large margin

Retargetability Assessment: User-study based approach

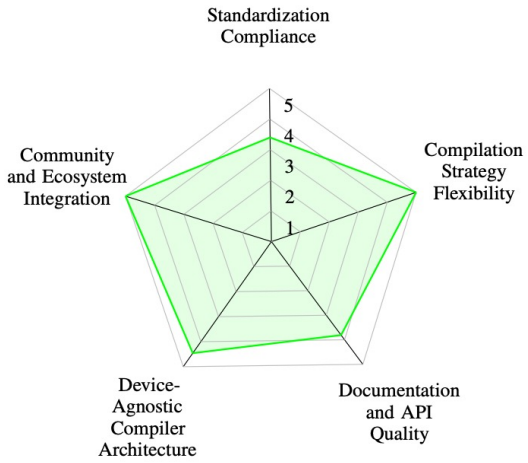
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q4-11
Tket	**P1**	3	1	1	4	4	3	3	3	4	3	5	
	P2	5	1	1	5	5	4	5	5	4	4	2	
	P3	3	1	1	5	4	4	5	5	4	3	4	
	P4	5	1	1	4	4	5	4	5	4	2	3	
	P5	2	1	1	5	5	5	5	5	5	5	5	
	P6	4	2	1	5	5	5	5	5	4	3	5	
		3.67	1.17	1	4.67	4.5	4.33	4.5	4.67	4.17	3.33	4	**4.27**
Qiskit	**P1**	3	3	2	4	4	4	3	4	2	4	5	
	P2	5	4	2	4	4	4	4	5	4	4	5	
	P3	3	4	1	5	5	5	5	5	4	5	5	
	P4	5	5	4	5	3	4	4	5	3	4	2	
	P5	2	2	1	4	4	3	4	2	4	3	1	
	P6	4	3	1	4	4	3	3	3	4	3	1	
		3.67	3.5	1.83	4.33	4	3.83	3.83	4	3.5	3.83	3.17	**3.81**
ProjectQ	**P1**	3	1	1	2	1	2	2	2	2	2	5	
	P2	5	1	1	1	1	1	2	3	2	1	5	
	P3	3	1	1	1	1	1	1	1	1	1	1	
	P4	5	1	1	1	1	2	2	4	2	1	1	
	P5	2	1	1	3	3	2	4	1	2	2	2	
	P6	4	1	1	1	1	1	1	3	2	1	5	
		3.67	1	1	1.5	1.33	1.5	2	2.33	1.83	1.33	3.17	**1.88**

- For Tket, hints were rated as least necessary among the evaluated compilers
- Qiskit and ProjectQ were rated at the same level

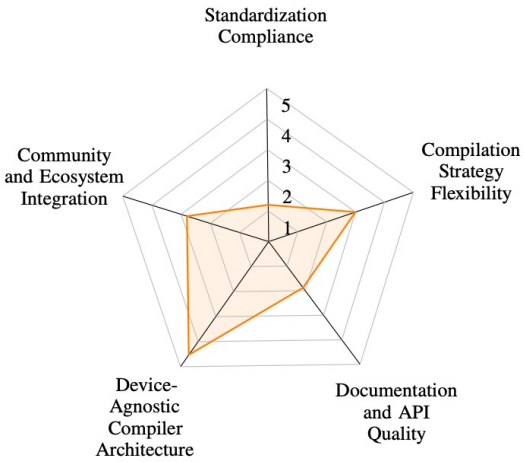
Retargetability Assessment: Overall Evaluation



(a) Tket



(b) Qiskit



(c) ProjectQ

- **Answering RQ2:** How retargetable are existing quantum compilers according to our methodology?
  - Tket demonstrates best retargetability, with score of **4.65**
  - Qiskit demonstrates strong retargetability, with score of **4.33**
  - ProjectQ shows weakest retargetability, with score of **2.68** (with strong score in Device-Agnostic Compiler Architecture)

Conclusion

- Proposed **methodology** to **measure retargetability** of quantum compilers over five dimensions:

Compilation Strategy Flexibility

Standardization Compliance

Device-Agnostic Compiler Architecture

Community and Ecosystem Integration

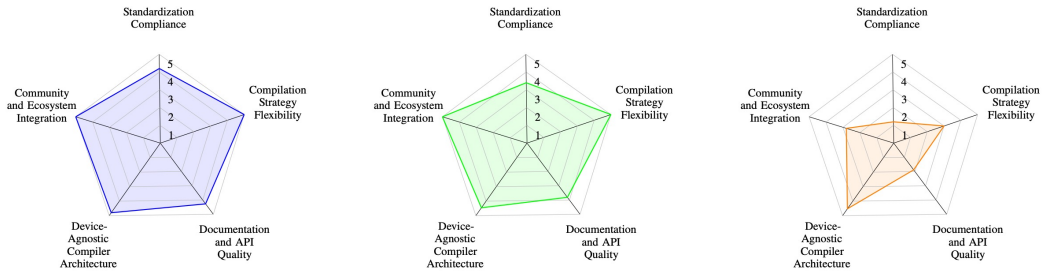
Documentation and API Quality

Evaluation via metric

Evaluation via study

- Evaluated retargetability of Tket, Qiskit and ProjectQ with proposed methodology:
  - Tket performs best (**4.65**), followed by Qiskit (**4.33**)
  - ProjectQ performs worst in comparison (**2.68**)

- Research could be expanded by evaluating more compilers, e.g. Weaver, which was released after study execution



(a) Tket (b) Qiskit (c) ProjectQ