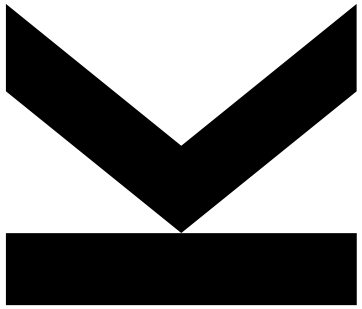


SPECIFYING FEATURE-DEPENDENT MTAINTAINABILITY REQUIREMENTS IN AN OPERATIONAL MANNER – CASE STUDY WITH PRACTITIONERS

Philipp Haindl and Reinhold Plösch

IWSM-MENSURA 2020, **Oct. 30, 2020**

OVERVIEW OF THE TALK

- Research Context and Motivation
- TAICOS Approach and Constraint DSL
- Case Study
  - Research Questions
  - Design and Participants' Tasks
- Results
- Conclusion and Future Work

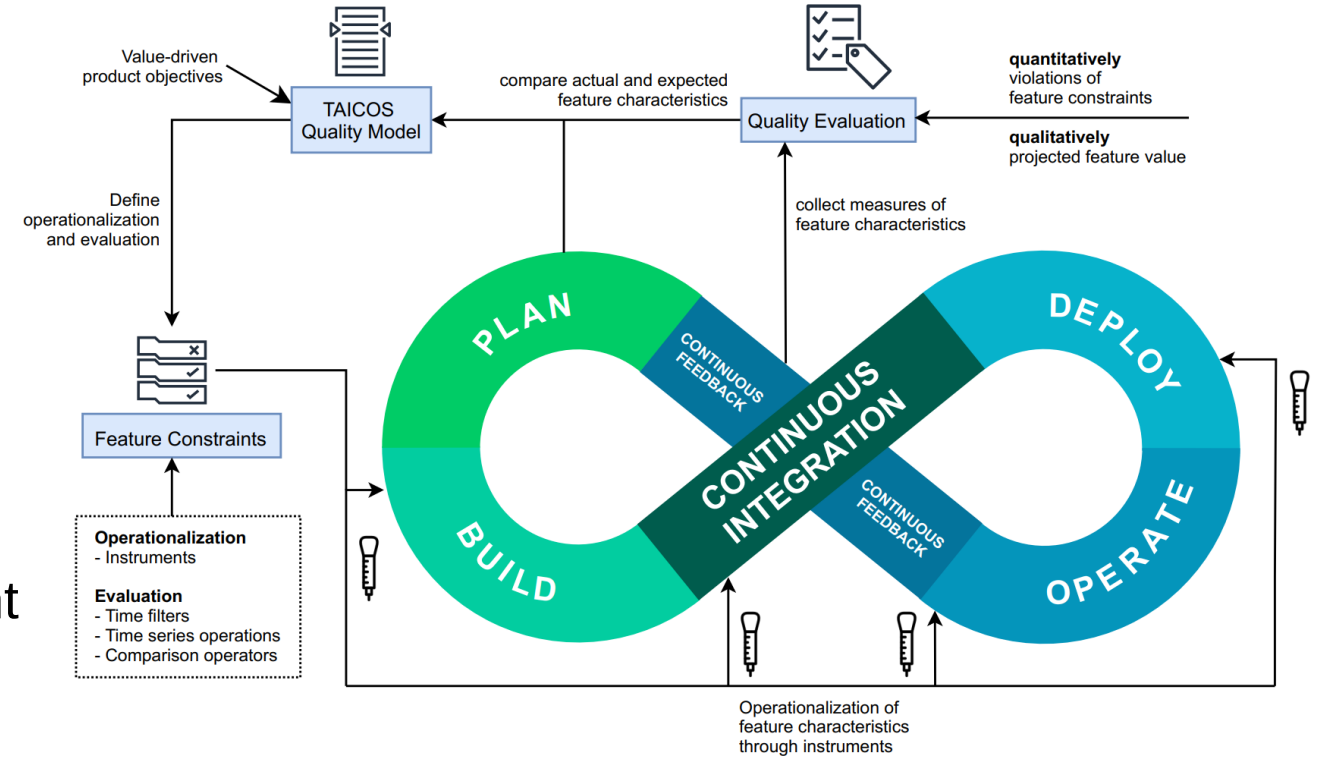
RESEARCH CONTEXT AND MOTIVATION

- Non-functional requirements differ among software features
- **Example:** Performance requirements on your running watch, maintainability requirements of critical vs seldomly used features
- Lack of operational definition and evaluation of these NFRs on feature level
- Practice: many projects use the most critical value of an NFR and demand fulfillment
  - → increased engineering effort which doesn't pay back and isn't required

- **TAICOS approach for specifying, measuring, and evaluating feature-dependent NFRs in DevOps**

TAICOS APPROACH

- **Specifying**
  - Operational Constraints DSL
  - Formalize NFRs as constraints
- **Measuring**
  - Acquire measures through instruments
  - → Software prototype
- **Evaluating**
  - Criteria to evaluate constraint fulfillment
  - → Software prototype



TAICOS – CONSTRAINT LANGUAGE

Acquisition of Measures

- Acquire the measures from the different systems
- Types of acquired data
  - Measures (numerical)
  - Ratings (ordinal)
  - Rule (numerical)
  - Benchmark (ordinal)

```
bugs as measure from Sonar("bugs").
vulnerabilities as measure from Sonar("vulnerabilities").
reliabilityRating as rating from Sonar("reliability_rating").
technicalDebt as measure from Sonar("sqale_index").
duplications as measure from Sonar("duplicated_lines","relative=lines").
cyclomaticComplexity as measure from Sonar("complexity","relative=lines").
documentPublicMethods as rule from Sonar("muse-java:DocumentPublicAbstractMethods")
commentedOutCodeLines as rule from Sonar("squid:CommentedOutCodeLine").
```

variable: commentedOutCodeLines — type: rule — instrument: Sonar("squid:CommentedOutCodeLine")

Evaluation of Feature Constraints

- Evaluate constraint expressions
- Time series operations
  (avg, min, max, median, gradient)
- Comparison operators *(<, <=, >=, >)*
- Time filters *(days, weeks, months)*
- Threshold value *(numeric, ordinal)*

```
cluster_feature, search_feature: documentPublicMethods < 120.
common_feature: documentPublicMethods < 200.

cluster_feature, search_feature: documentPublicMethods.benchmark <= Q50.
common_feature: documentPublicMethods.benchmark <= Q75.
```

feature: common_feature — variable: documentPublicMethods — threshold: Q75

```
cluster_feature: avg(cyclomaticComplexity, days(180)) < 20.
search_feature: avg(cyclomaticComplexity, days(180)) < 10.
```

operation: avg(cyclomaticComplexity, days(180)) — time filter: days(180)

CASE STUDY

CASE STUDY: GOAL AND RESEARCH QUESTIONS

- **Goal:** Determine if practitioners can apply the language for expressing maintainability requirements of their companies (→ Case Study with practitioners)
- **Research Questions:**
  - Are scope and expressiveness of the different language elements appropriate to specify feature-dependent NFRs?
  - Is the combination of the different language elements suitable to express maintainability requirements?
  - What are typical language patterns depending on the maintainability requirements in the companies
  - What are benefits and weaknesses of the language?

CASE STUDY: DESIGN AND TASKS

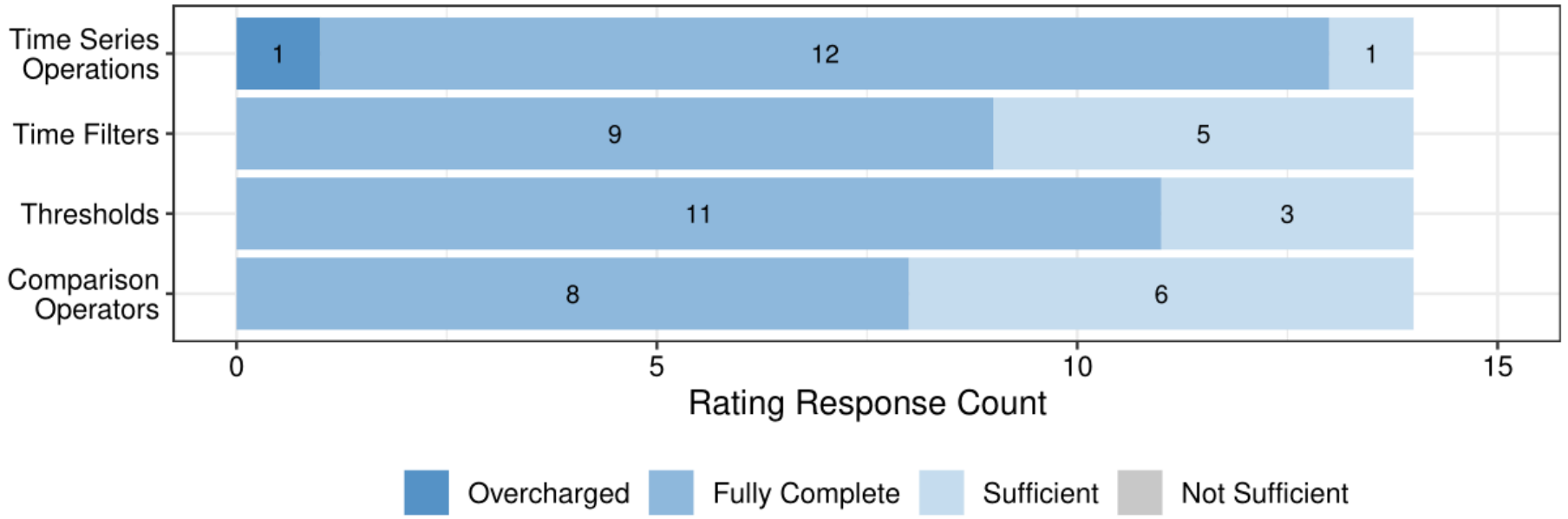
- 14 domain experts, i.e., (senior) software engineers, DevOps engineers, and architects each with > 10 years of experience
- **Case study design**
  - *Introductory part:* Detailed explanation of the constraint language, Q&A with participant, explanation of purpose, sequence, and concrete tasks in the case study
  - *Practical part:* Participants choose 3 features and 5 maintainability requirements of their companies and apply the constraint language
  - *Interview part* (10 open, 13 closed questions):
    - Assessing scope, expressiveness, suitability of the language elements on a 4-point Likert scale
    - Missing operators and time filters
    - Problems and weaknesses, questions regarding visualization of constraint evaluation results

RESULTS

- **Quantitative analysis**
  - 222 constraint (basis to assess frequency of data types, time series operators, time filters)
- **Qualitative analysis**
  - Codified expressed maintainability requirement of the constraints
  - Condensed 43 categories (in total) into 9 main categories of maintainability requirements
  - 397 statements (28 on average per interview)

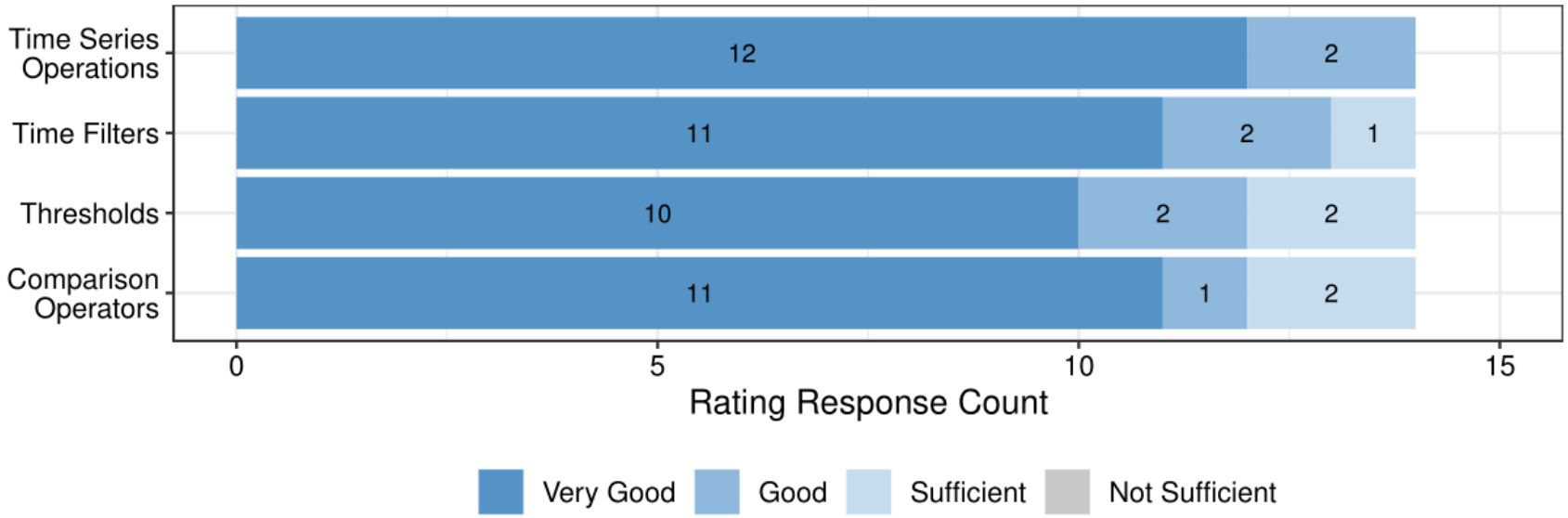
RESULTS – SCOPE OF LANGUAGE ELEMENTS

- In total, very positive feedback
- Two important observations: assessment of time series operations (overcharged, sufficient)



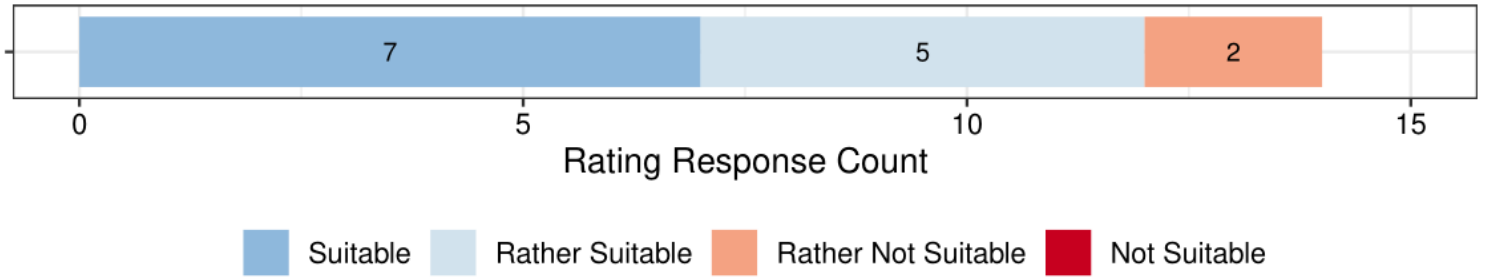
RESULTS – EXPRESSIVENESS OF LANGUAGE ELEMENTS

- Again, very positive feedback, no particular observations



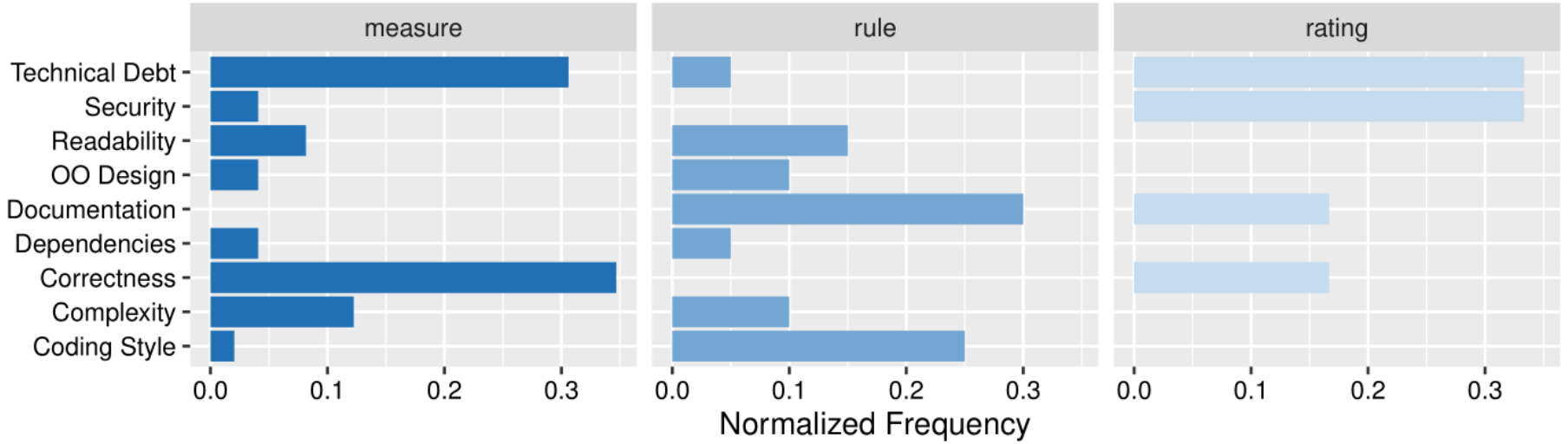
RESULTS – SUITABILITY

- All ratings at least suitable
- Reasons for two participants rating as *rather not suitable*
  - Modularization of features in their prevents definition of meaningful constraints
  - Due to large number of features in their companies: essential for them to rather define general constraints for all features and only exceptions



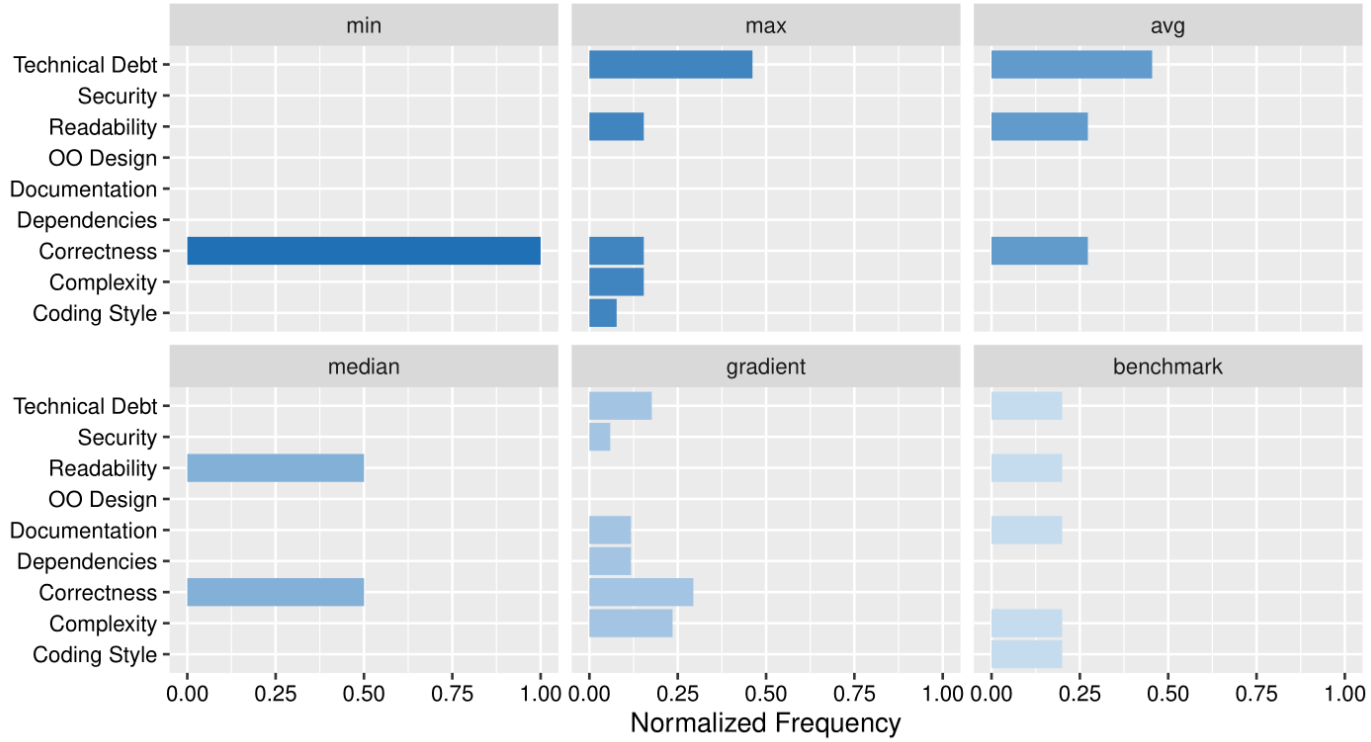
RESULTS – USAGE PATTERNS PER MAINTAINABILITY ASPECT (DATA TYPES)

- **Total instances:** 49 measure, 20 rule, 6 rating
- *Measure* most heavily used among all types



RESULTS – USAGE PATTERNS PER MAINTAINABILITY ASPECT (OPERATIONS)

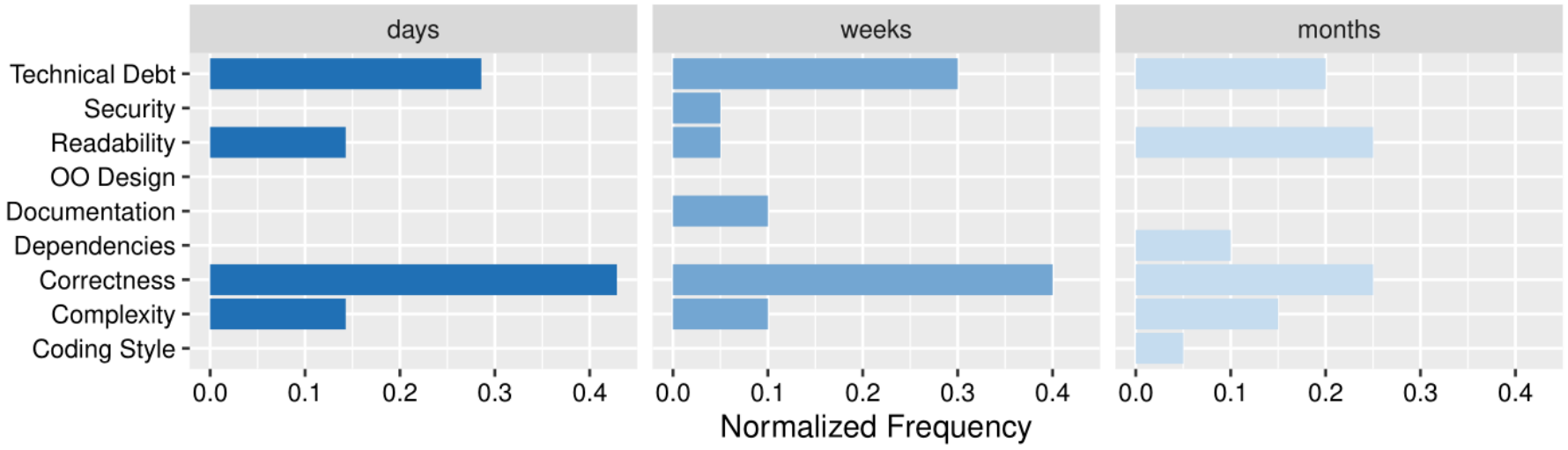
- **Total instances:** 5 min, 13 max, 11 avg, 2 median, 17 gradient, and 5 benchmark operations
- Gradient and benchmark operations not as frequently used as expected



RESULTS – USAGE PATTERNS PER MAINTAINABILITY ASPECT (TIME FILTERS)

Total instances: 7 days, 20 weeks, and 20 months

Time filter usage equally distributed



CONCLUSION AND FUTURE WORK

- In all aspects, majority rates completeness, scope, and expressiveness on high scores
- **Limitations:**
  - Notion of *feature* in companies and how to apply TAICOS notion of it
  - Large quantity of features
  - In same companies, industry sector, no time series are interesting: e.g., when no rule violations are allowed at all
- **Future work**
  - Methodological support (which metrics are when important)
  - Set operations, advanced comparison operations, advanced time filters (for sprints/iterations, or events)
  - Tool support, when faced with a large number of constraint specifications

JOHANNES KEPLER
UNIVERSITY LINZ

Q&A

Philipp Haindl

Insitute of Business Informatics – Software Engineering
Johannes Kepler University Linz, Austria
T +43 732 2468 4258
E philipp.haindl@jku.at

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at