Microservices in the German Industry

Insights into Technologies, Characteristics, and Software Quality

Jonas Fritzsch, University of Stuttgart
Justus Bogner, University of Applied Sciences Reutlingen
Stefan Wagner, University of Stuttgart
Alfred Zimmerman, University of Applied Sciences Reutlingen

Funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program ***Services Computing***
http://www.services-computing.de/?lang=en

21.02.2019

Motivation

- Popularity of service-oriented computing
- Microservices emerged from industry practice
- Academia
  - Early adopters and “pure” Microservices
  - Diversity in industry practice [7, 8]
  - Academic assumptions may be incorrect for typical systems [7]

→ Find reasons for peculiarities in industry

→ In-depth interviews with software professionals

Scope

- Research objective:

  *Provide insights into **industry adoption** and **implementation** of **Microservices** as well as into **rationales** in this area*

- RQ1: Which **technologies** do companies use for the implementation and operation of Microservices and with what rationale?
- RQ2: Which **characteristics** of Microservices do companies neglect and for what reasons?
- RQ3: How do companies perceive the influence of Microservice architectures on **software quality**?

Research Method

- Semi-structured interviews
- Participants
  - Software professionals with significant experience
  - Recent participation in the development of a service-based system
- Interviews of ~45 to ~70 minutes (audio conferencing / face to face)
- Audio-Recording and creation of transcripts
- Case characterization matrix and cross-case analysis
- Online repository with artefacts and results
  https://github.com/xJREB/research-microservices-interviews

Interview Demographics

- 10 companies
    - Various sizes (from 1-25 up to >100,000 employees)
    - Different domains (e.g. *SW & IT Services* or *Retail*)
- 17 participants
    - Technical roles (e.g. architect or developer)
    - Considerable experience (mean of ~15 years)
- 14 systems
    - Various sizes (from 6 up to ~250 services)
    - Different domains (e.g. *Automotive* or *Retail*)

Company	Domain	Employees	Participant	System
C1	Financial Services	1 - 25	P1: Developer	S1
C2	Software & IT Services	>100,000	P2: Lead Architect P3: Architect P4: Architect	S2 S3 S4
C3	Software & IT Services	26 - 100	P5: Architect P6: Lead Architect	S5
C4	Software & IT Services	101 - 1,000	P7: Architect P8: Architect	S6 S7
C5	Software & IT Services	>100,000	P9: Lead Developer	S8
C6	Tourism & Travel	1,001 - 5,000	P10: Developer P11: Data Engineer P12: Architect	S9 S10
C7	Logistics & Public Transport	101 - 1,000	P13: DevOps Engineer P14: Architect	S11
C8	Retail	5,001 - 10,000	P15: Lead Architect	S12
C9	Software & IT Services	101 - 1,000	P16: Architect	S13
C10	Retail	1,001 - 5,000	P17: Lead Architect	S14

ID	System Purpose	Inception	People	Services
S1	Derivatives management	Rewrite	7	9
S2	Freeway toll management	Rewrite & Extension	10	10
S3	Automotive problem management	Rewrite & Extension	50	10
S4	Public transport sales	Rewrite & Extension	~300	~100
S5	Business analytics	Greenfield	7	6
S6	Automotive configuration mgmt	Rewrite	20	60
S7	Retail online shop	COTS Replacement	~200	~250
S8	IT service monitoring platform	Continuous Evolution	15	9
S9	Hotel search engine	Continuous Evolution	~50	~10
S10	Hotel management suite	Rewrite & Extension	50	20
S11	Public transport management suite	Continuous Evolution	~175	10
S12	Retail online shop	COTS Replacement	~85	~45
S13	Automotive end-user services mgmt	Rewrite & Extension	30	7
S14	Retail online shop	COTS Replacement	~350	~175

Results: Service Technology (RQ1)

- RESTful HTTP
  - Used in all 14 cases
  - Rationales: interoperability, technolo
  - Sometimes seen as harmful coupling (P5, P6, P15)
- Docker containers
  - Used in 11 cases, planned for 3 cases
  - Rationales: operability, orchestration, portability
  - But: brings additional complexity, especially with Kubernetes

P6: *“We also have some REST-based communication between services, which is not 100% clean.”*

Results: Service Technology (RQ1)

- Java
    - Us
    - Rationales:
    - Large Java parts would lead to synergies
- Single Page Applications (SPAs)
    - Used in 9 cases (e.g. Angular, Vue, React)
    - Rationales: rich and desktop-like UIs, communication with REST APIs
    - But: may lead to client-side logic or high response times

> P12: *“Java is not generally bad for me, but if you choose Java, take it for everything. Then it’s awesome. But mixing it with something else is completely the contrary of awesome.”*

Results: “Pure” Microservices? (RQ2)

Characteristics by Lewis & Fowler [1] :

- Componentization via Services
- Organized around Business Capabilities
- Products not Projects
- Smart endpoints and dumb pipes
- Decentralized Governance & Data Management
- Infrastructure Automation & DevOps
- Design for failure
- Evolutionary Design

Results: “Pure” Microservices? (RQ2)

- DevOps practices & automation
  - Only 5 cases relied on *“You build it, you run it.”*
  - Very different degrees of automation
  - Only 3 cases reli
- Servi
  -
  - Often la
  - Rationales: performance, dependencies, domains that are hard to cut

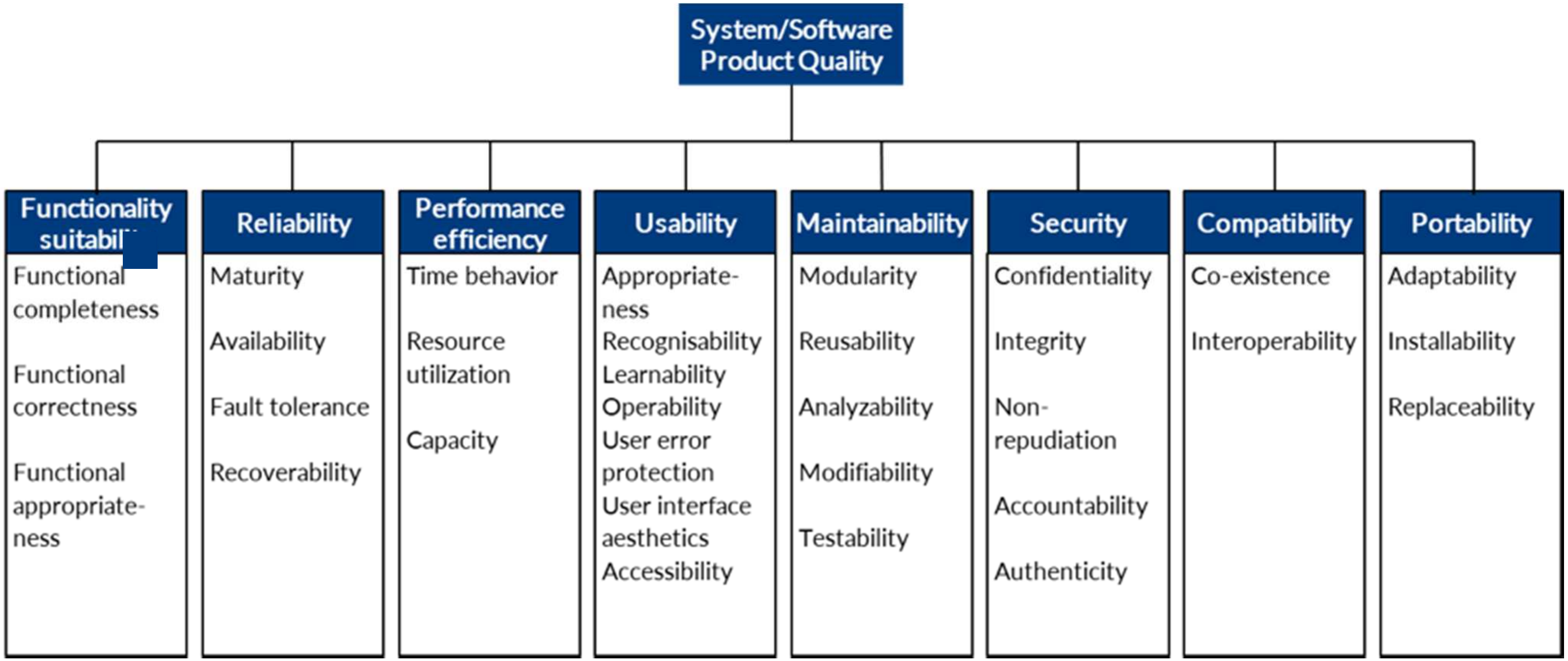
P7: *“If we went all the way with Microservices, we probably would have to create a separate service for each business domain entity. That would be too much, we can’t go that route.”*

Results: “Pure” Microservices? (RQ2)

- Products, not proje
  - Syst
  - Inv
- Decentralizatio
  - Higher degree of governance for external systems
  - Very few companies went “all in” (C6, C8, C10)
- Technological heterogeneity
  - Generally less diversity, even though the possibility is left open
  - Seen controversially: highly valued or perceived as dangerous

P4: *“In our case, the main challenge is to convince 300 people to move in the same direction. For that, we created a very large amount of guidelines and rules for service creation.”*

Results: Impact on SW Quality (RQ3)

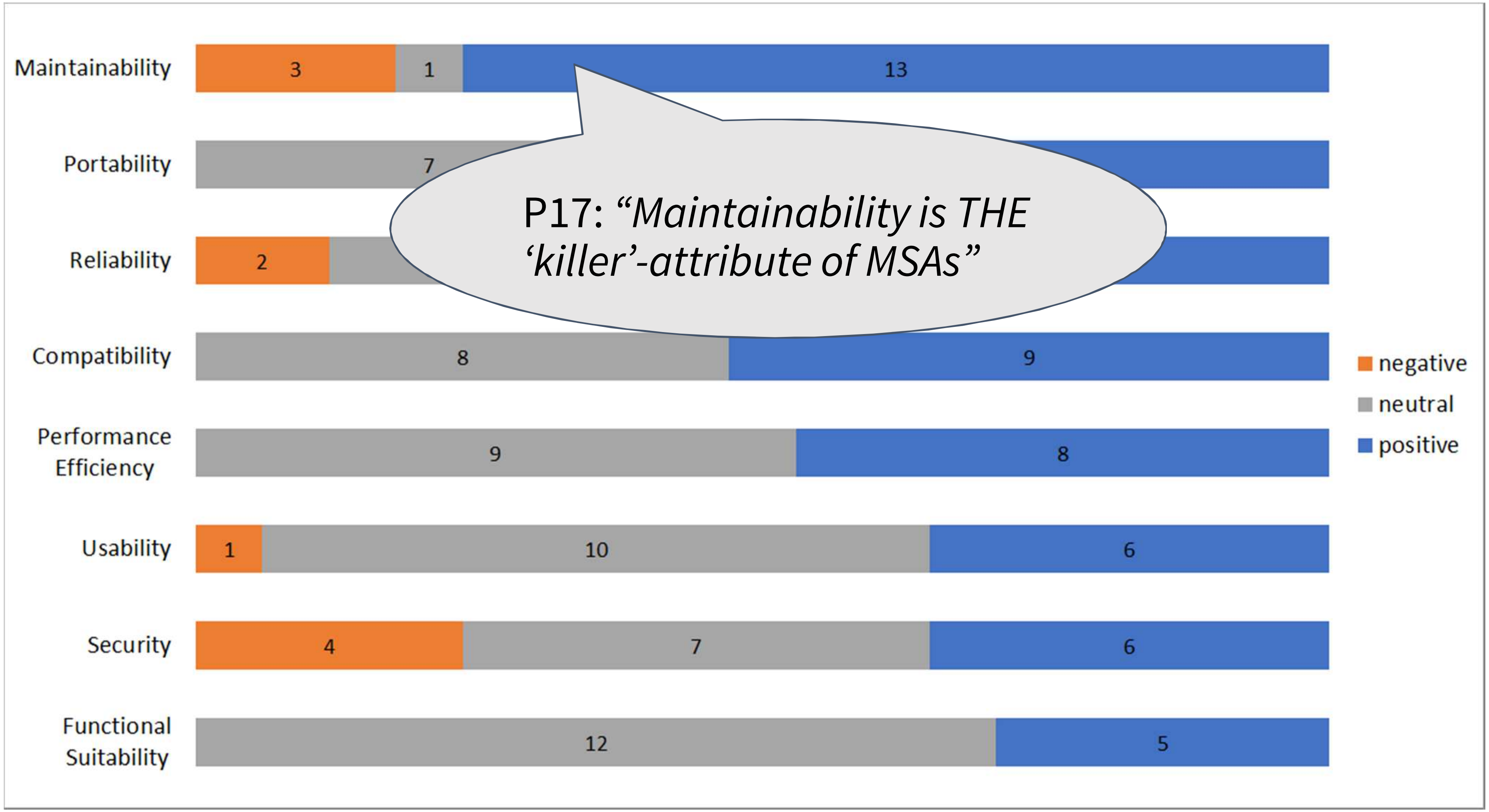


System/Software Product Quality

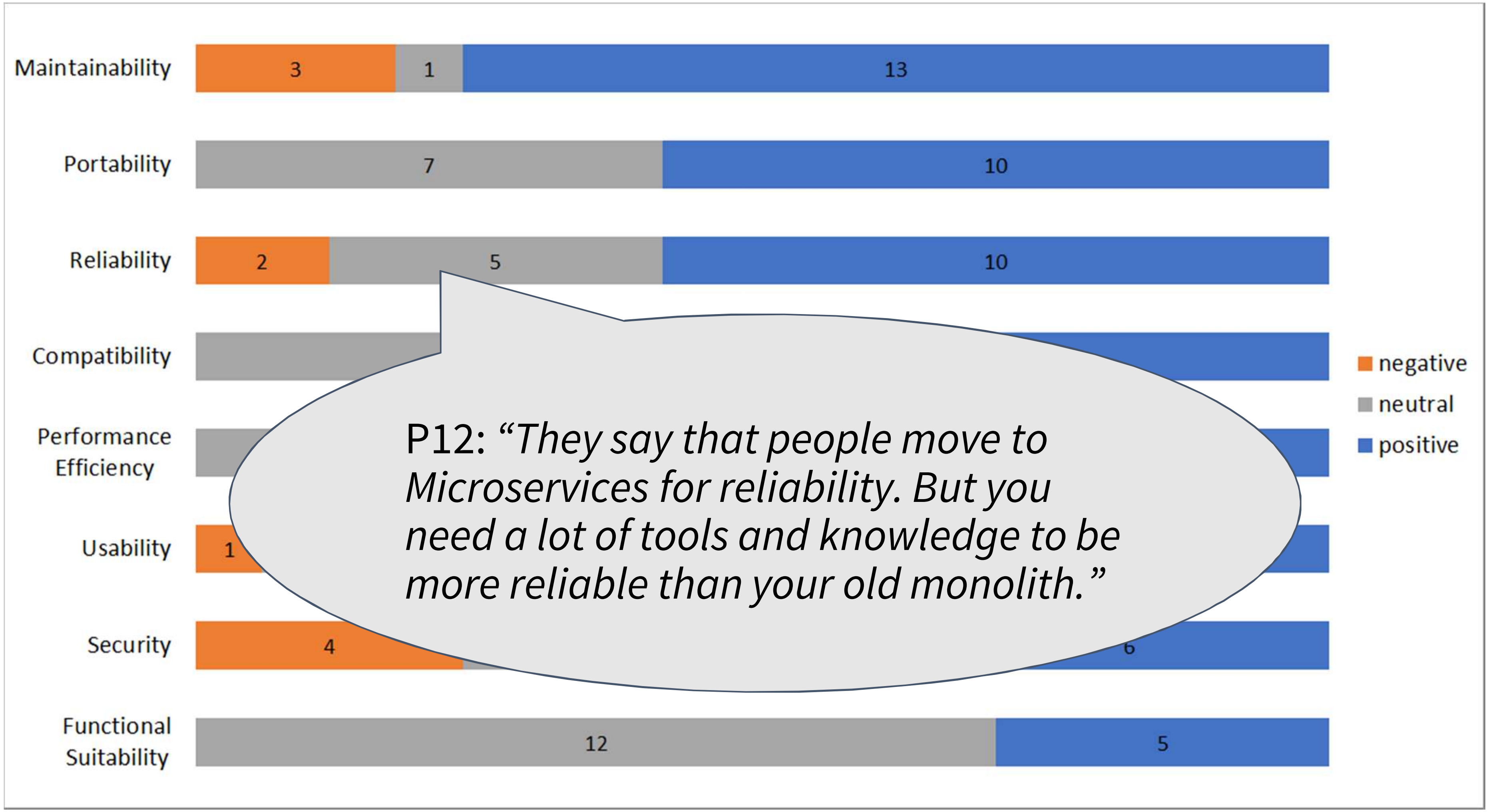
Functionality suitabil	Reliability	Performance efficiency	Usability	Maintainability	Security	Compatibility	Portability
Functional completeness	Maturity	Time behavior	Appropriateness	Modularity	Confidentiality	Co-existence	Adaptability
Functional correctness	Availability	Resource utilization	Recognisability	Reusability	Integrity	Interoperability	Installability
Functional appropriateness	Fault tolerance	Capacity	Learnability	Analyzability	Non-repudiation		Replaceability
	Recoverability		Operability	Modifiability	Accountability		
			User error protection	Testability	Authenticity		
			User interface aesthetics				
			Accessibility				

https://www.inovex.de/blog/operationalisierung-des-begriffs-digitale-qualitaet-und-entwicklung-von-messansaetzen/

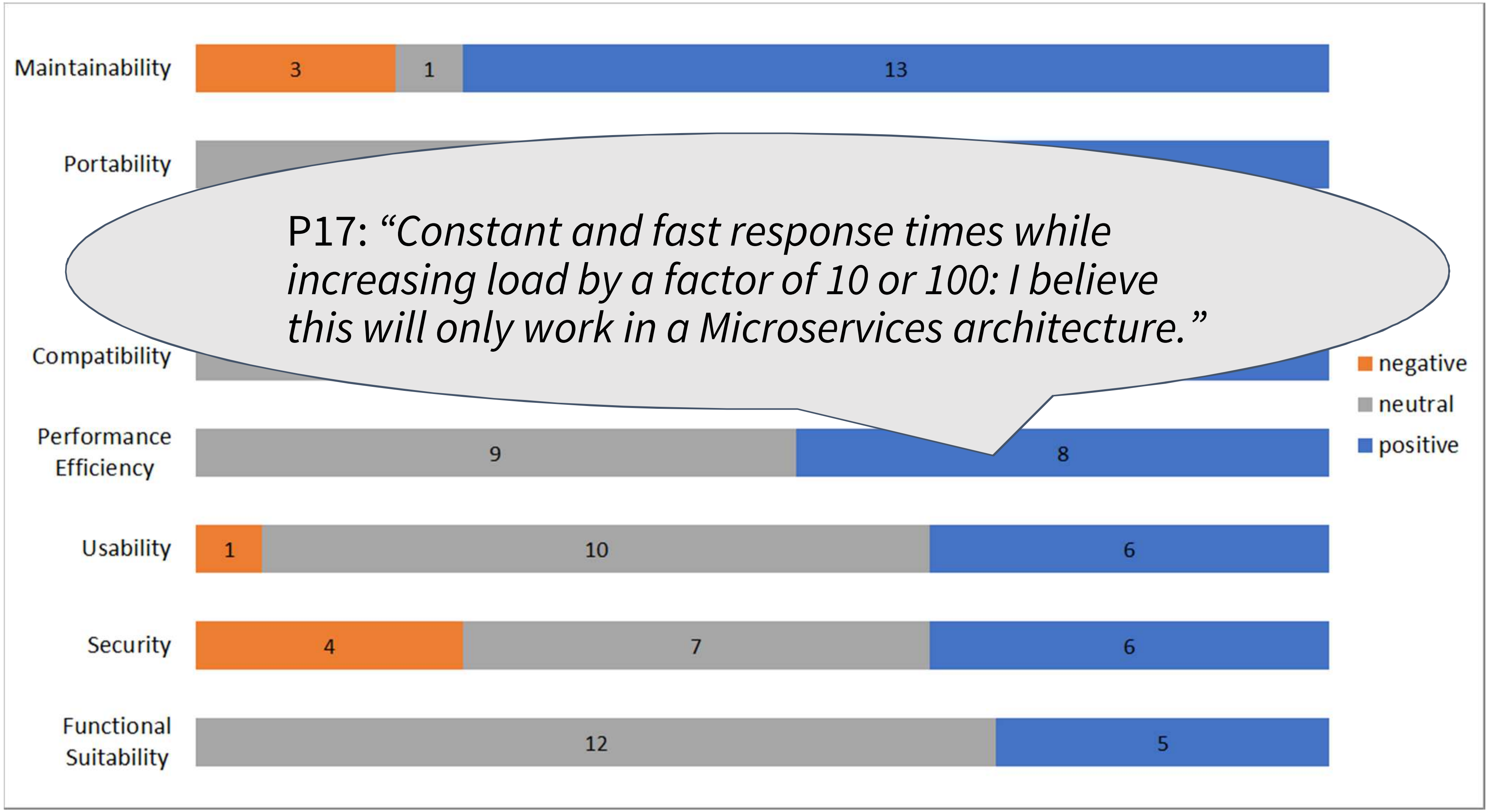
According to ISO/IEC 25010



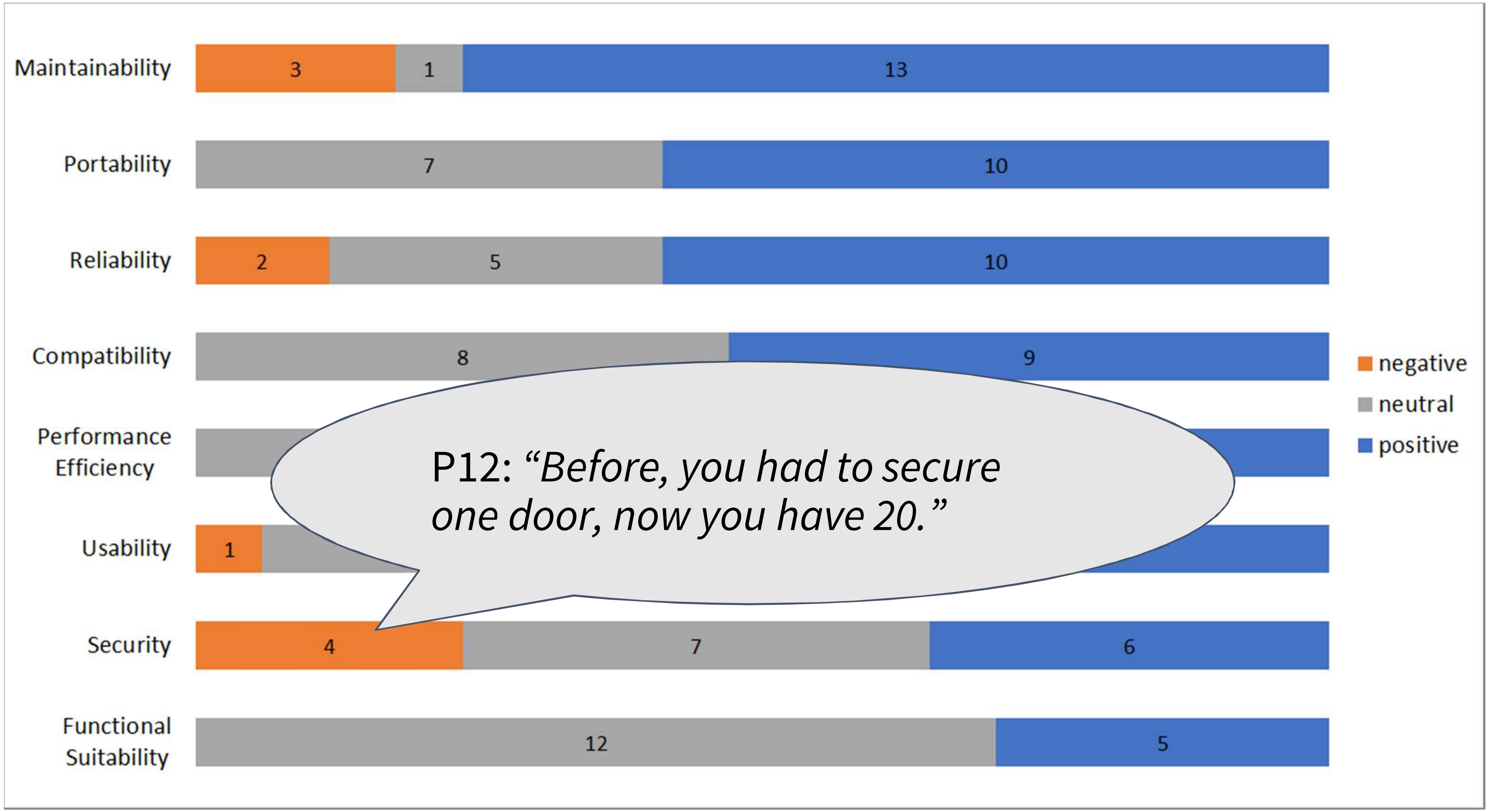
P17: *“Maintainability is THE ‘killer’-attribute of MSAs”*



P12: *“They say that people move to Microservices for reliability. But you need a lot of tools and knowledge to be more reliable than your old monolith.”*



P17: *“Constant and fast response times while increasing load by a factor of 10 or 100: I believe this will only work in a Microservices architecture.”*



P12: *“Before, you had to secure one door, now you have 20.”*

Threats to Validity

- Internal validity
  - Participants may not have revealed their true opinions
  - Participants may have misunderstood questions or concepts
  - Researcher bias may affect Interpretation validity
- External validity
  - No generalization for distributions possible (with only 14 cases)
  - Participants exclusively based in Germany
  - ~52% of participants from software and IT services companies

Conclusion and Implications

- RESTful HTTP and Docker containers prevalent (but not without critique)
- Microservices as a scale, not a binary switch
- Internal use vs. external customer
- Tendency for fewer and more coarse-grained services
- Line between service- and Microservice-based systems is blurry
- Positive or neutral impact on software quality

⇒ Future industry-focused research should take this into account

Thank you!

Q & A