HOW TO GET A HANDLE ON THESE SLIPPERY QUALITY REQUIREMENTS?

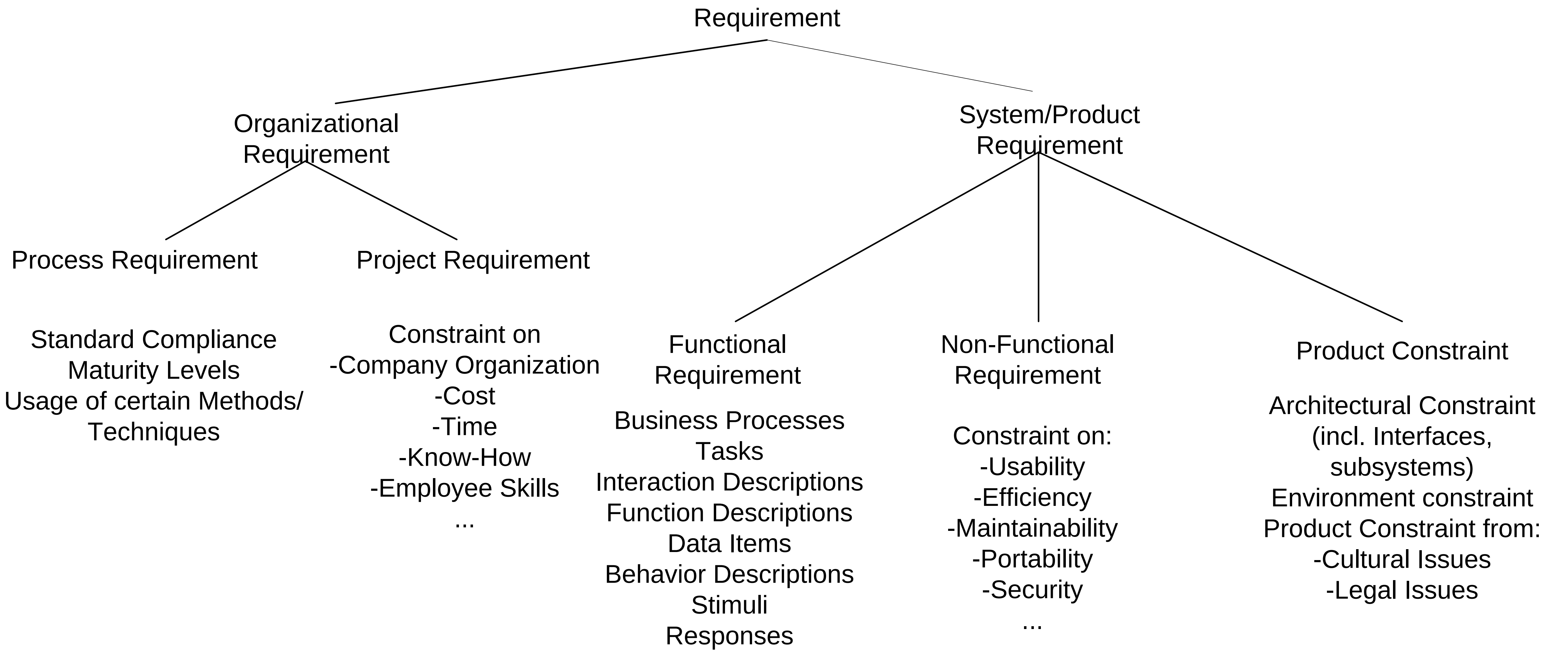
Dr. Joerg Doerr, Division Manager „Information Systems“

Fraunhofer IESE, Kaiserslautern, Germany

Agenda

- My understanding of QR
- Why is it so hard to tackle QR
- Which best practices worked for traditional? Which don't?
- NFR in agile settings
- Opportunities of agile and DevOps settings
- New approaches make it to industry!

Requirements Taxonomy



Requirement

Organizational Requirement

Process Requirement

Standard Compliance
Maturity Levels
Usage of certain Methods/ Techniques

Project Requirement

Constraint on
-Company Organization
-Cost
-Time
-Know-How
-Employee Skills
...

System/Product Requirement

Functional Requirement

Business Processes
Tasks
Interaction Descriptions
Function Descriptions
Data Items
Behavior Descriptions
Stimuli
Responses

Non-Functional Requirement

Constraint on:
-Usability
-Efficiency
-Maintainability
-Portability
-Security
...

Product Constraint

Architectural Constraint (incl. Interfaces, subsystems)
Environment constraint
Product Constraint from:
-Cultural Issues
-Legal Issues

Quality Models

- There are **many quality models** for software / system qualities
  - ISO 9126
  - Further development of 9126 into ISO 250010 Series
  - Qualities of the NFR-Framework (Chung, Mylopulous, etc.)
  - …
- Quality attributes also appear in the typical requirements specification standards (IEEE830, IEEE1362, Volere, ...)

Types of Quality Attributes / Requirements (ISO 25010)

- Quality in Use (relative to human use)
  - Effectiveness
  - Efficiency
  - Satisfaction
  - Freedom of Risk
  - Context Coverage

- Product Quality (intrinsic)
  - Functional Suitability
  - Performance Efficiency
  - Compatibility
  - Usability
  - Reliability
  - Security
  - Maintainability
  - Portability

Very important distinction for agile: **run-time** attributes (like response time, usability, …)
vs. **development-time** attributes (maintainability, portability, …)

Why is it so hard to deal with NFRs?

- Technical Reasons
  - Often **tacit**
  - **Too many methods** (specific ones) around to deal individual ones
  - …
- Political Reasons
  - **No motivation** of the customer to be specific and write them down
  - **No motivation** of contractor to be specific…
- But on the other side:
  - They are the real **architecture breakers**
  - They **spoil** complete **contracts**

10 Best Practices in dealing with NFRs

1. Separate project specific **NFR** and project-spanning **quality attributes**!
2. Specify **measurable** NFR
3. Differentiate the NFRs according to the **reference objects** they refer to (system, data, tasks, …)
4. **Focus** the **effort** for NFR elicitation
5. Specify NFR as close as possible to the FRs
6. Use checklists and tools for the elicitation
7. Identify all relevant stakeholders and involve them in the elicitation sessions
8. Analyze NFR dependencies early on
9. Document the **rationales** for NFR
10. Establish a systematic process to deal with NFR

How did those practices work in general?

- **Always positive ROI** when applying these best practices (**ROI** between **2** and **17**)
- Between **120% and 620% more relevant NFRs** identified

10 Best Practices in dealing with NFRs – how did they work?

1. Separate project specific NFR and project-spanning quality attributes! √
2. Specify measurable NFR √ → **>95%** of the NFRs were measurable
3. Differentiate the NFRs according to the reference objects they refer to (system, data, tasks, …) √ → **significant reduction of complexity**
4. Focus the effort for NFR elicitation 🛇 → customer wants everything, prioritization is difficult
5. Specify NFR as close as possible to the FRs √
6. Use checklists and tools for the elicitation √
7. Identify all relevant stakeholders and involve them in the elicitation sessions 🛇
8. Analyze NFR dependencies early on ⚡ → complex task!
9. Document the rationales for NFR √ → really important! We got rid of many unnecessary NFR!
10. Establish a systematic process to deal with NFR 🛇 → process change takes time

Product Quality in agile setting vs. traditional

- **Our observations: is agile worse than traditional in terms of product quality?**
  - In general: no
  - **But:**
  - Many times in **very early** shipments: yes!
  - But mid-term better balance between FR and NFR, because of early and many releases
  - Typically less overfulfillment of NFRs
  - But “surprise” NFRs at later stages become architecture breakers (e.g., if in early shipments, no full load is imposed to a system, performance deficiencies hide)
- **Run-time vs. development-time** matters!
  - Run-time √
  - Development time ⚡

Where do NFR hide in agile development settings?
→ Theory

- Independent User Stories
- Acceptance criteria
- Definition of Done / Definition of Ready
- Test cases / Regression Testing
- NFR in Scaled Agile Framework
  - NFRs as Backlog Constraints / extra columns in product backlog
  - Suggestion of FURPS categorization (Functionality, Usability, Reliability, Performance, and Supportability)

Image taken from https://www.romanpichler.com/

Constraints on the backlog
NFRs
© Scaled Agile, Inc.

Where do NFR “hide” in agile development settings?
→ Practice

- Orally (feedback-sessions)
- Bug-tracking system
  (in product backlog together with the user stories)
- UX-concepts
- Architecture-Documents
- Spike-Solutions

- Introduction of “technical product owner” / explicit architect

So which best practices can we use for agile settings?

- Some best practices can be directly applied, others not

  → understand and transfer the most important **basic concepts**, e.g.:

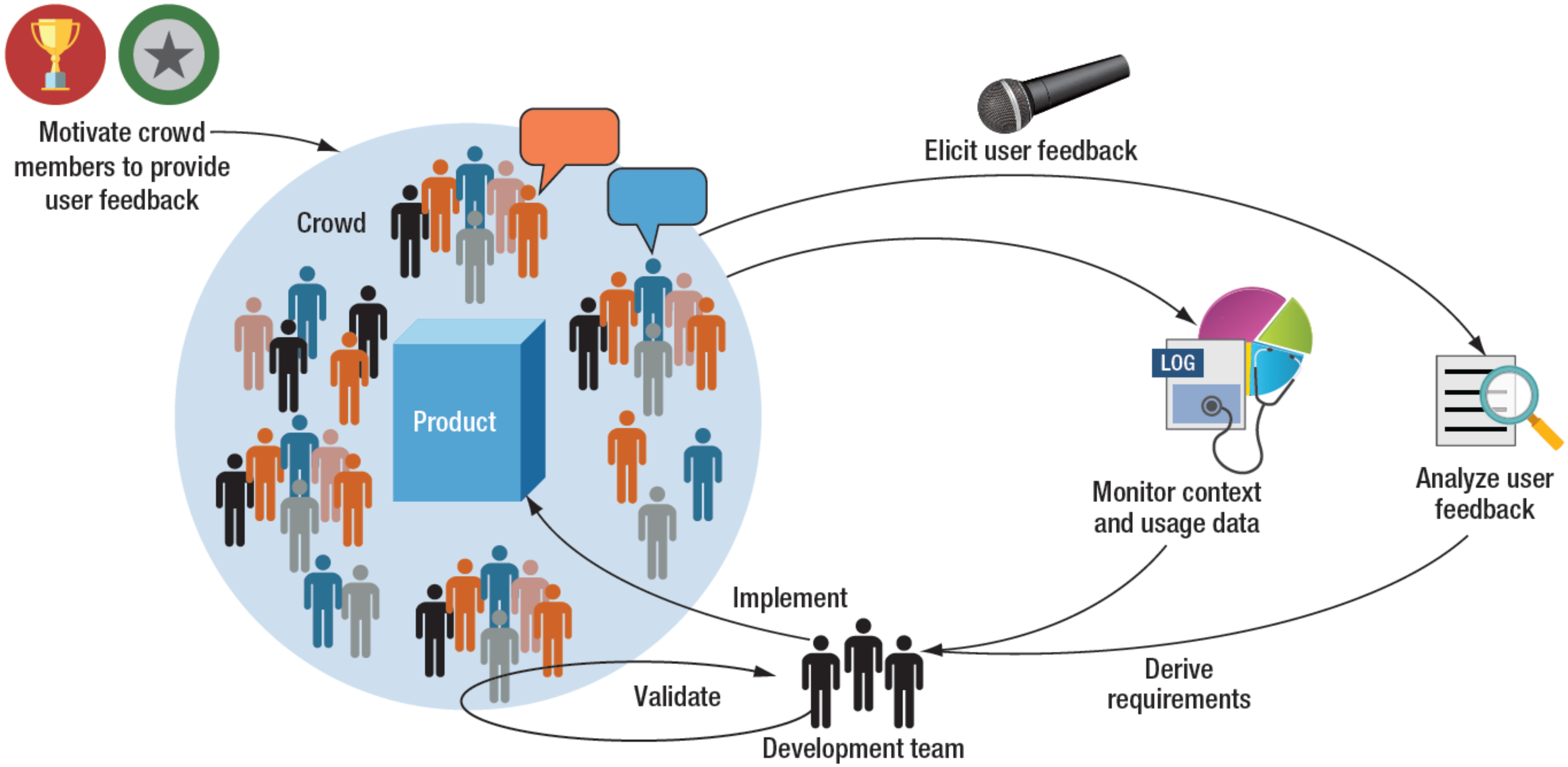
  - Separate NFR and QA
  - Classification of quality attributes wrt. reference-objects is essential and **guides your way**:
    - task related NFR: use acceptance criteria
    - data related NFR: use independent user story
    - system related NFR (especially development-time attributes): use backlog constraint
  - Insist on **rationale** (also for acceptance criteria)
  - **Measurable** works also for the agile ones!
  - Use **dedicated sessions with checklists for the elicitation**
  - …

But there is more to it: Opportunities of Agile and DevOps settings

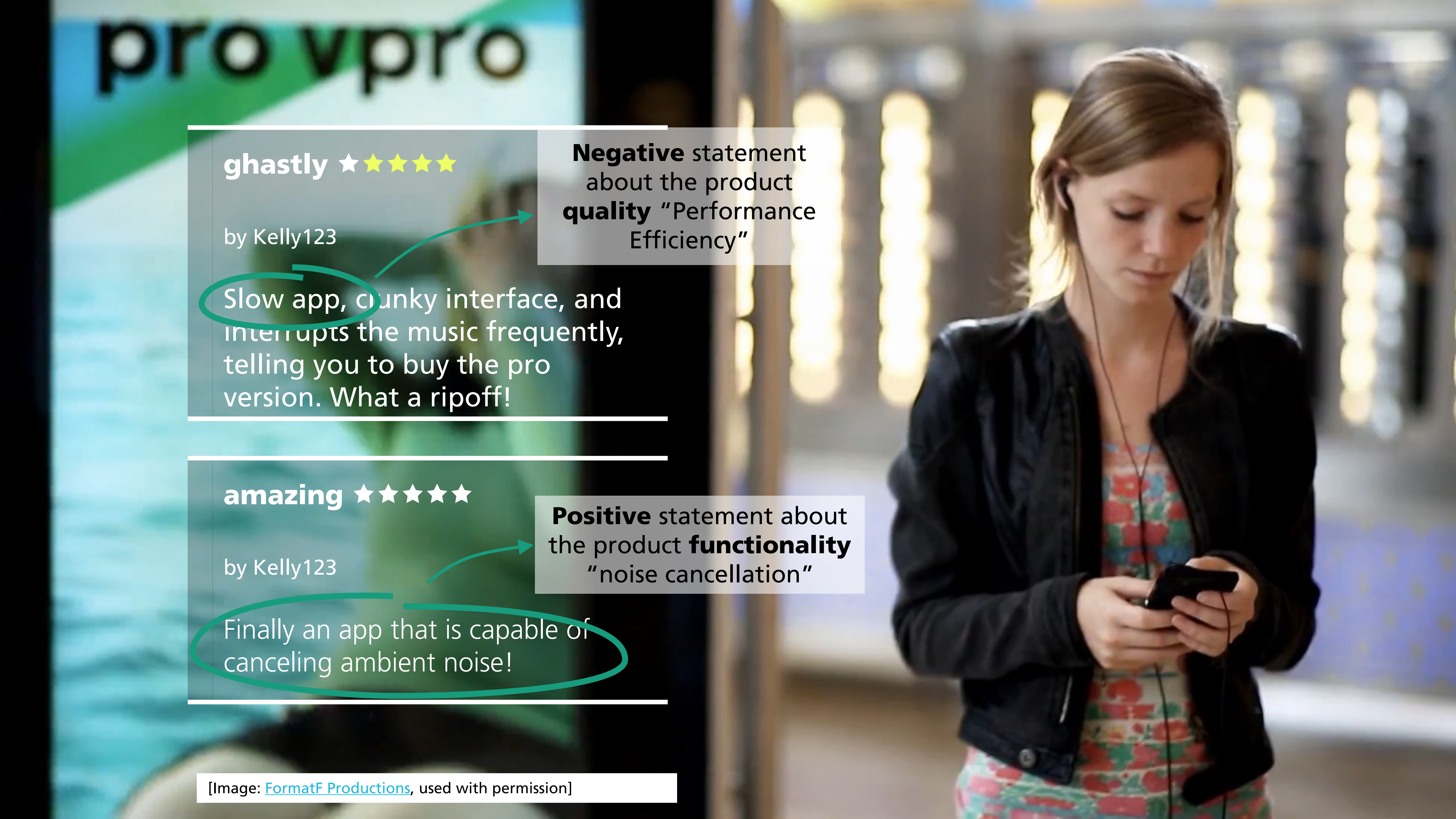
- Agility and especially DevOps settings provide the possibility to obtain **early feedback** – let’s make use of it

→ New approaches appear not only in research, they now **make it to industry**!

The Crowd-based RE Approach



Source: Groen, E. C., Seyff, N., Ali, R., Dalpiaz, F., Doerr, J., Guzman, E., Hosseini, M., Marco, J., Oriol, M., Perini, A., & Stade, M. (2017). The crowd in requirements engineering: The landscape and challenges. IEEE Software, March/April 2017. Image © 2017 IEEE Society.



ghastly ★★★★★

by Kelly123

Slow app, clunky interface, and interrupts the music frequently, telling you to buy the pro version. What a ripoff!

Negative statement about the product **quality** “Performance Efficiency”

amazing ★★★★★

by Kelly123

Finally an app that is capable of canceling ambient noise!

Positive statement about the product **functionality** “noise cancellation”

[Image: FormatF Productions, used with permission]

Requirements Elicitation from User Feedback

- RE research increasingly focuses on online **user feedback**, e.g.,
  - Feature requests
  - Praise / criticism
  - Bug reports

} Strong focus on functional aspects

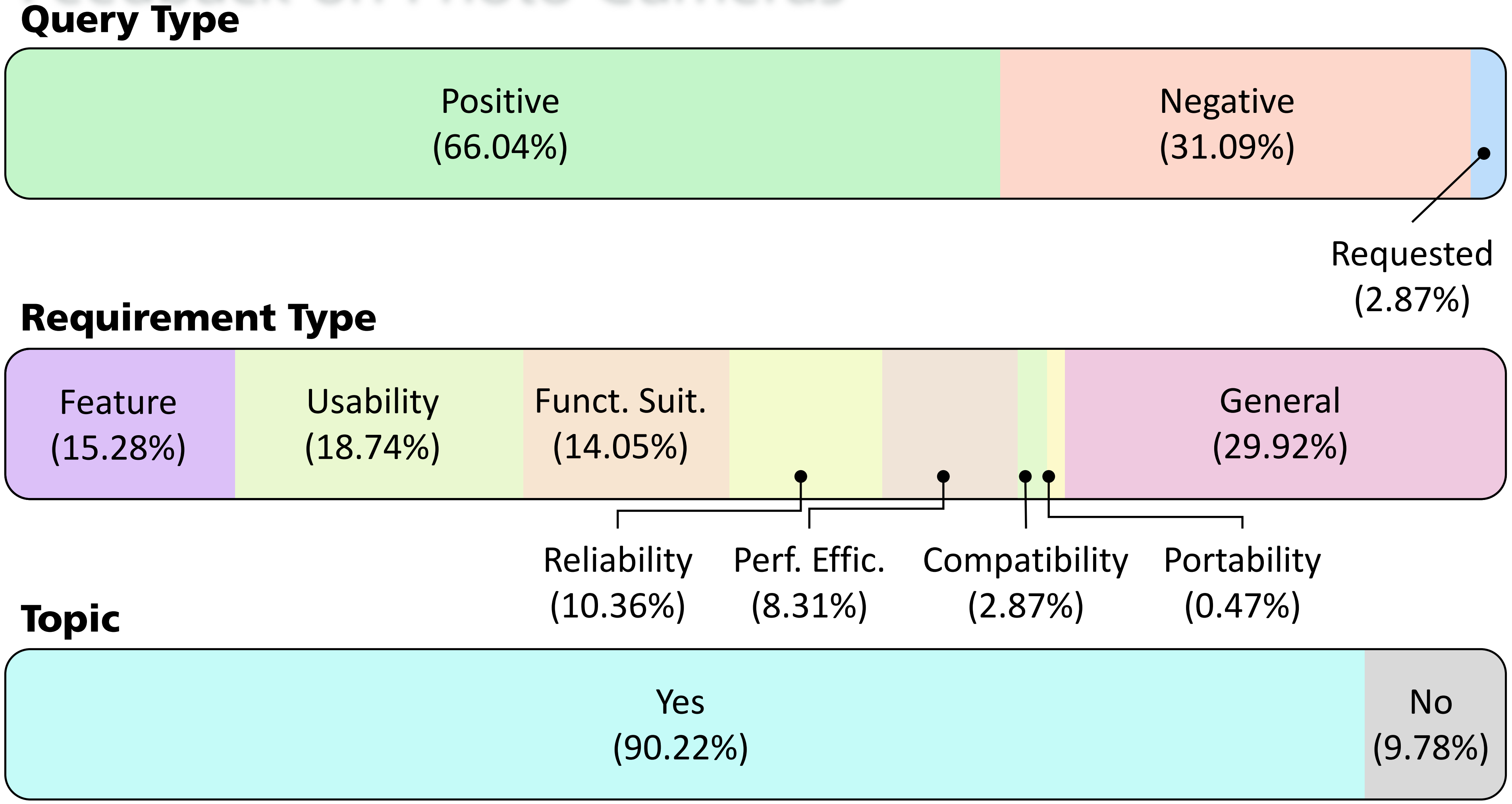
← Mostly: feature shortcomings on a functional level

- Online user feedback is widely available on platforms such as:
  - App stores
  - Social media
  - Issue tracking systems
  - Internal channels of companies

→ Due to the effort involved in manual analysis, most solutions classify user feedback using **text mining** approaches

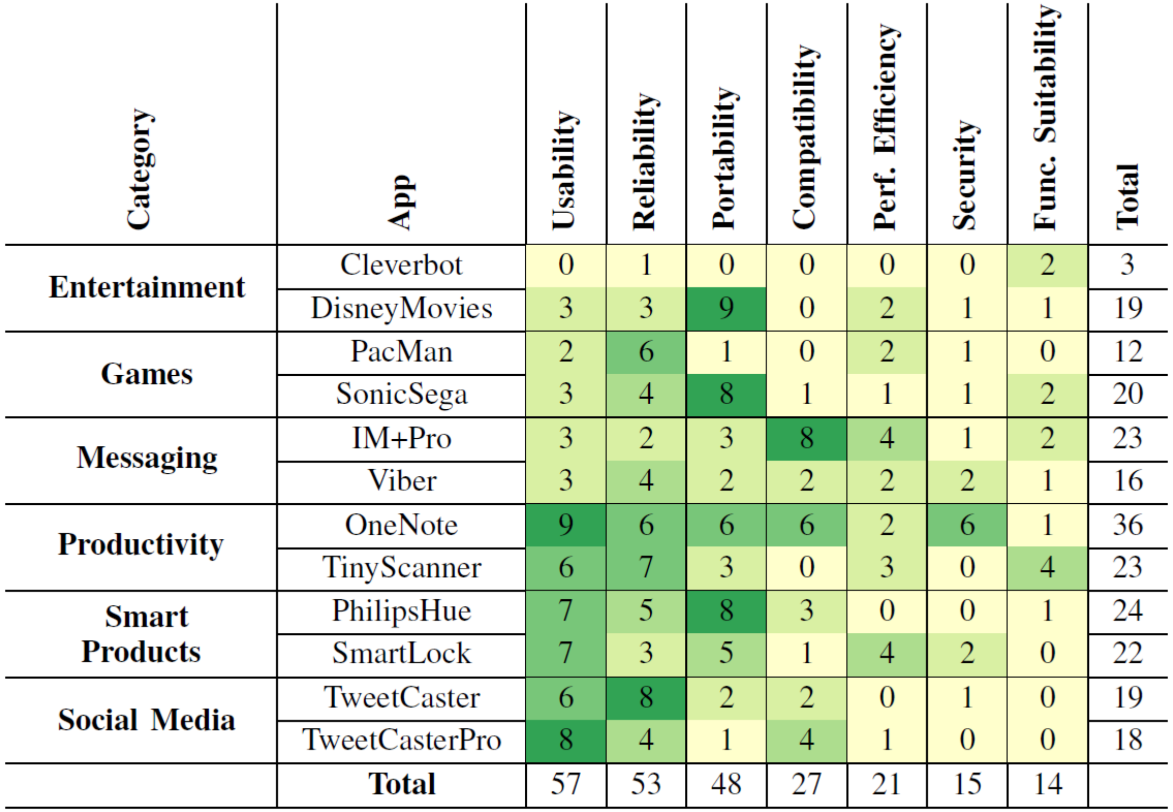
Classification of Statements

User Feedback on Photo Cameras



Quality Analysis

Quality Aspects Addressed



Category	App	Usability	Reliability	Portability	Compatibility	Perf. Efficiency	Security	Func. Suitability	Total
Entertainment	Cleverbot	0	1	0	0	0	0	2	3
	DisneyMovies	3	3	9	0	2	1	1	19
Games	PacMan	2	6	1	0	2	1	0	12
	SonicSega	3	4	8	1	1	1	2	20
Messaging	IM+Pro	3	2	3	8	4	1	2	23
	Viber	3	4	2	2	2	2	1	16
Productivity	OneNote	9	6	6	6	2	6	1	36
	TinyScanner	6	7	3	0	3	0	4	23
Smart Products	PhilipsHue	7	5	8	3	0	0	1	24
	SmartLock	7	3	5	1	4	2	0	22
Social Media	TweetCaster	6	8	2	2	0	1	0	19
	TweetCasterPro	8	4	1	4	1	0	0	18
	Total	57	53	48	27	21	15	14	

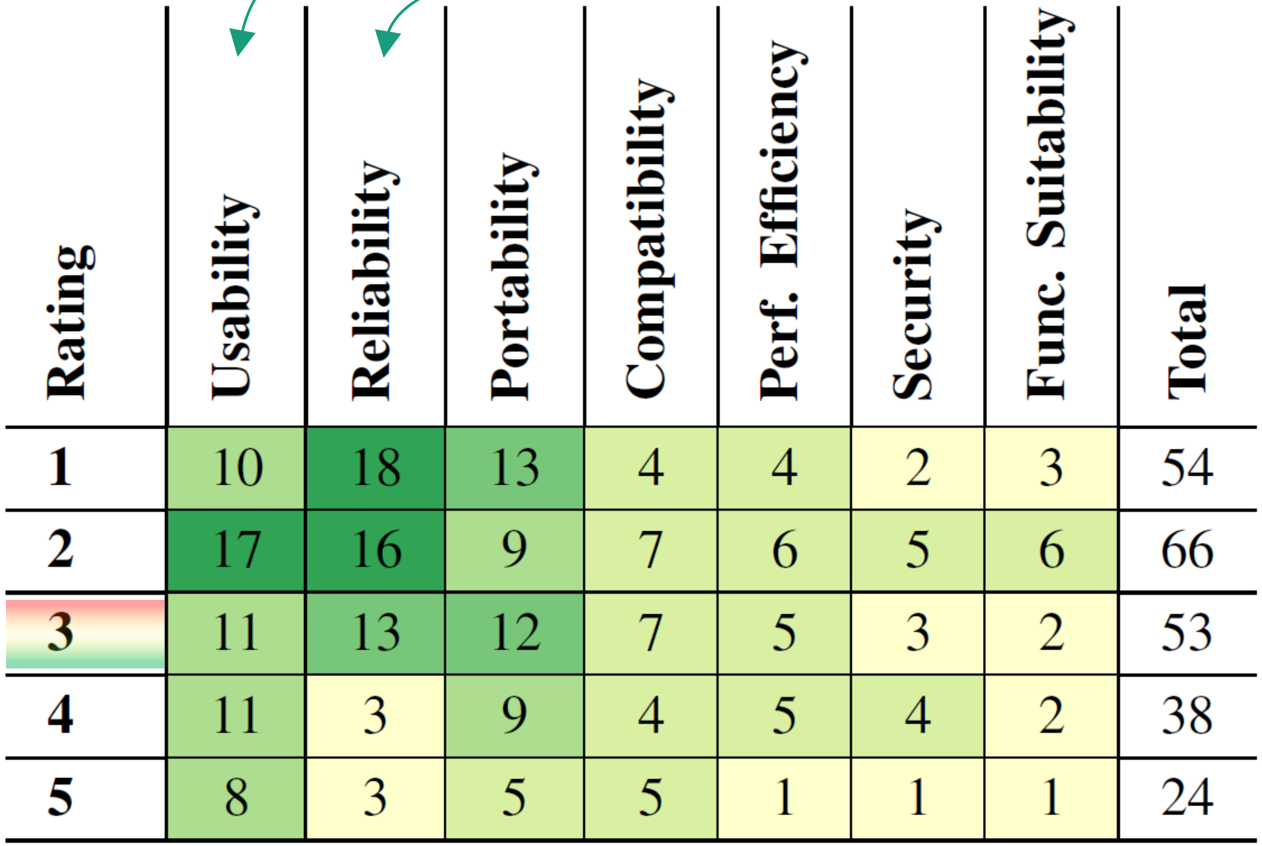
- Note: “Maintainability” was found for 0 apps

Quality Analysis

Distribution of Characteristics by Rating

Both positive and negative statements

Mostly in negative reviews

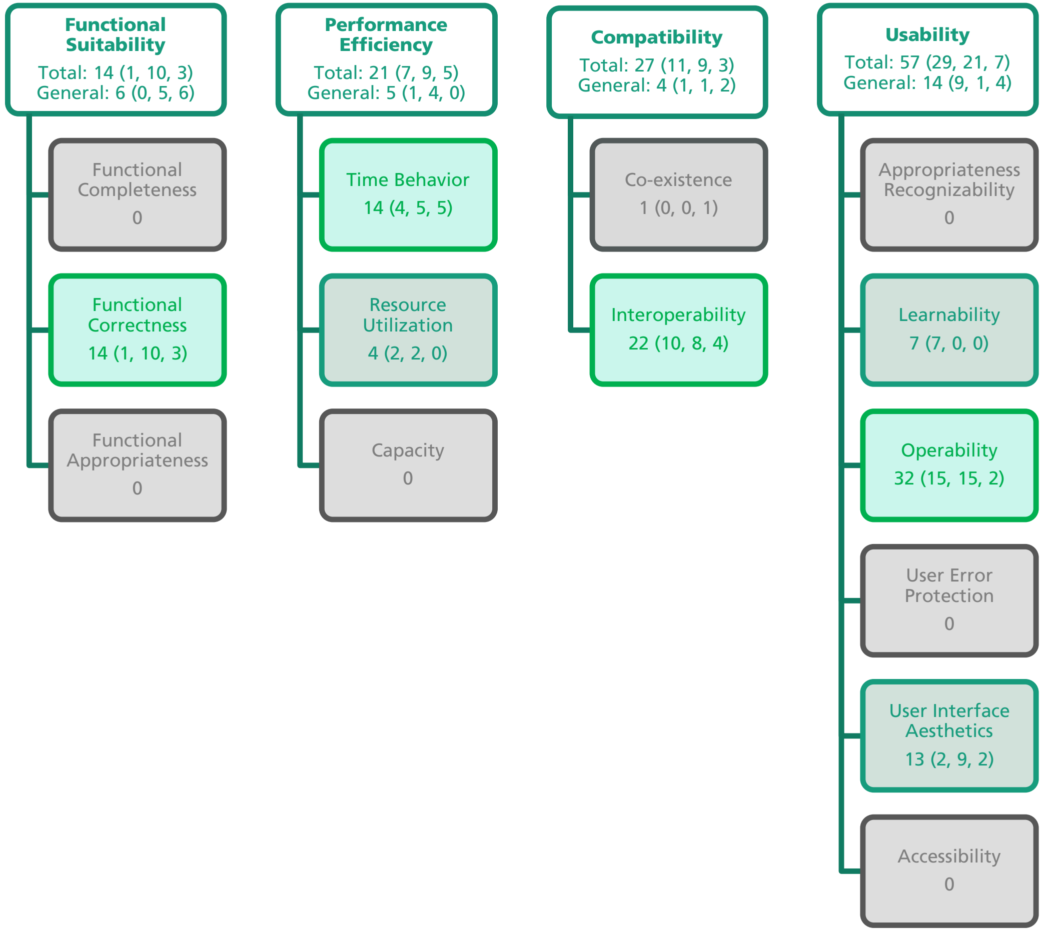


Rating	Usability	Reliability	Portability	Compatibility	Perf. Efficiency	Security	Func. Suitability	Total
1	10	18	13	4	4	2	3	54
2	17	16	9	7	6	5	6	66
3	11	13	12	7	5	3	2	53
4	11	3	9	4	5	4	2	38
5	8	3	5	5	1	1	1	24

Quality Analysis

Addressed ISO 25010 Subcharacteristics

- Frequency by number of reviews (max: 360) per abstore
- App stores:
  - Amazon.com
  - Apple AppStore
  - Google Play Store



Functional Suitability
Total: 14 (1, 10, 3)
General: 6 (0, 5, 6)

- Functional Completeness 0
- Functional Correctness 14 (1, 10, 3)
- Functional Appropriateness 0

Performance Efficiency
Total: 21 (7, 9, 5)
General: 5 (1, 4, 0)

- Time Behavior 14 (4, 5, 5)
- Resource Utilization 4 (2, 2, 0)
- Capacity 0

Compatibility
Total: 27 (11, 9, 3)
General: 4 (1, 1, 2)

- Co-existence 1 (0, 0, 1)
- Interoperability 22 (10, 8, 4)

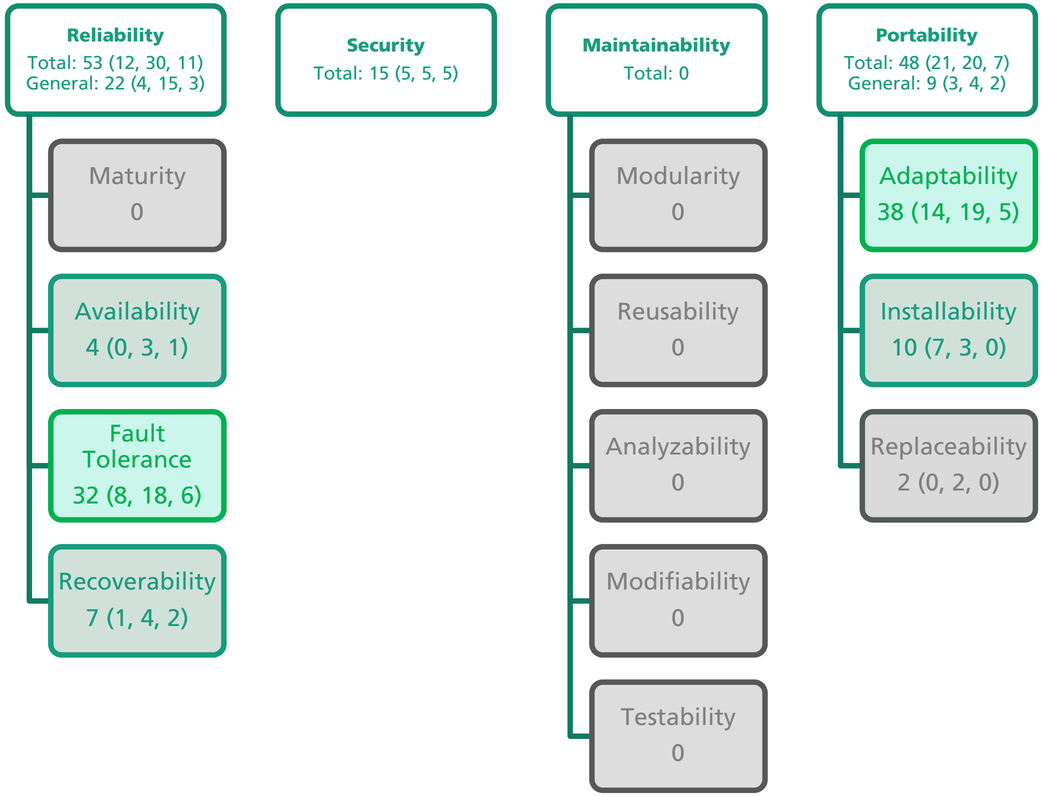
Usability
Total: 57 (29, 21, 7)
General: 14 (9, 1, 4)

- Appropriateness Recognizability 0
- Learnability 7 (7, 0, 0)
- Operability 32 (15, 15, 2)
- User Error Protection 0
- User Interface Aesthetics 13 (2, 9, 2)
- Accessibility 0

Quality Analysis

Addressed ISO 25010 Subcharacteristics

- Frequency by number of reviews (max: 360)
- App stores:
  - Amazon.com
  - Apple AppStore
  - Google Play Store



Reliability
Total: 53 (12, 30, 11)
General: 22 (4, 15, 3)

- Maturity 0
- Availability 4 (0, 3, 1)
- Fault Tolerance 32 (8, 18, 6)
- Recoverability 7 (1, 4, 2)

Security
Total: 15 (5, 5, 5)

Maintainability
Total: 0

- Modularity 0
- Reusability 0
- Analyzability 0
- Modifiability 0
- Testability 0

Portability
Total: 48 (21, 20, 7)
General: 9 (3, 4, 2)

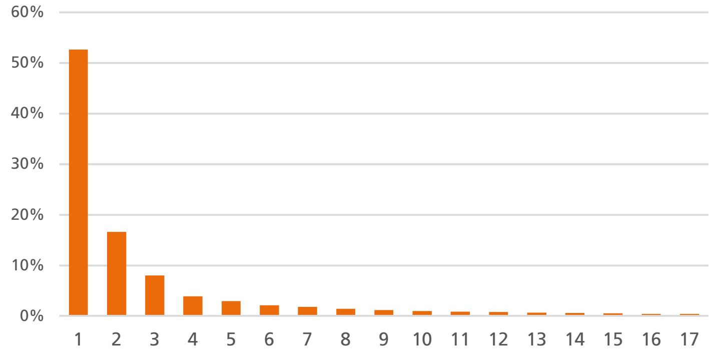
- Adaptability 38 (14, 19, 5)
- Installability 10 (7, 3, 0)
- Replaceability 2 (0, 2, 0)

Information from Usage Analysis - Example

- Examples for analysing usage analysis:
  - Websites: people running into dead-ends when performing tasks
  - Typical durations
  - Errors…
- In general: objective information on runtime attributes can be identified, NFRs can be derived
- Typically no development time attributes
- Industrial example in search engine data analysis
  - First hit must be of really high relevance
  - Quality of hit >=6 maybe not so critical

Usage Data

Search Engine Data Analysis



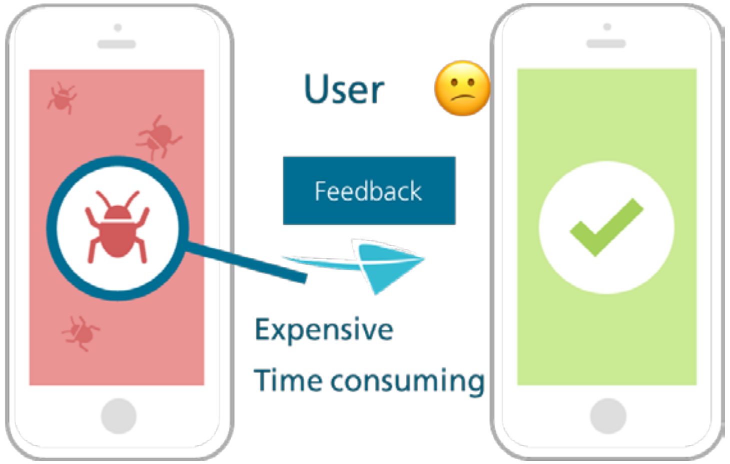
Integrated analysis of usage and text mining very promising research area

→ Before full-fledged, heavy-weight and expensive analysis procedures are used, often **more lightweight approaches** should be used

Doing Emoji Analysis

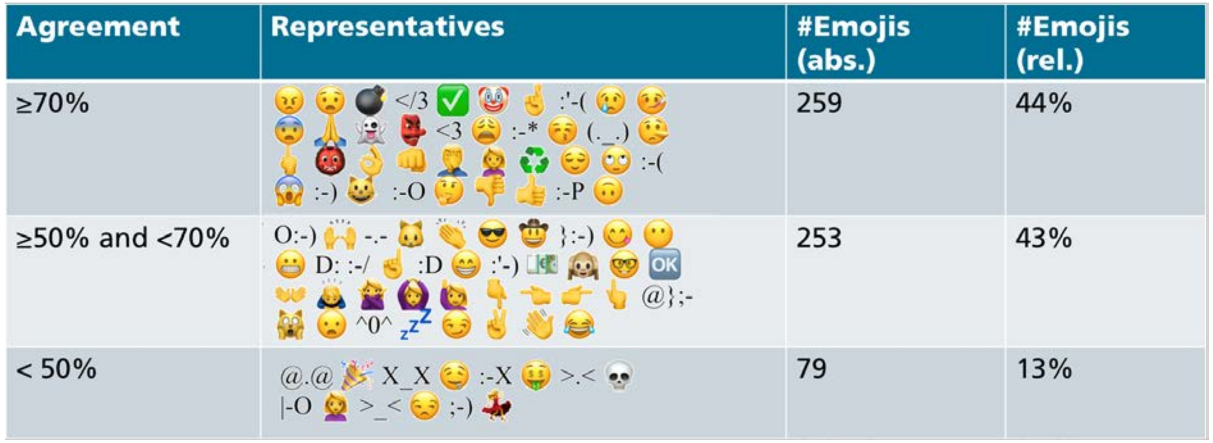
- Opti4Apps project uses **user feedback to improve (mobile) apps**
  - Focus is textual feedback
  - One aspect is to **use Emojis to gather emotions of users**

- We investigated how different Emojis are understood by people
  - Prerequisite for correct analysis of Emojis
- For this, we performed a poll in which many Emojis were classified by sentiment and emotion



Doing Emoji Analysis

- More than one hundred participants answered the first questionnaire
- Main result: People perceive Emojis **very homogeneously** (sentiment even better than emotion)!



Agreement	Representatives	#Emojis (abs.)	#Emojis (rel.)
≥70%	😠 😟 💣 </3 ✅ 🤡 🤞 :'-(😢 🤒 😨 🙏 👻 👺 <3 😩 :-* 😚 (._.) 🤥 👇 👹 👌 👊 🤦‍♂️ 🙍‍♀️ ♻️ 😌 🙄 :-(😱 :-) 😺 :-O 🤔 👎 👍 :-P 🙃	259	44%
≥50% and <70%	O:-) 🙌 -.- 🐱 👏 😎 🤠 }:-) 😋 😶 😬 D: :-/ ☝️ :D 😁 :'-) 💶 🙉 🤓 🆗 👐 🙇 🙅‍♀️ 🙆‍♀️ 🙋‍♀️ 👇 👈 👉 👆 @};- 🙀 ☹️ ^0^ 💤 😏 ✌️ 👋 😂	253	43%
< 50%	@.@ 🎉 X_X 🤤 :-X 🤑 >.< 💀 |-O 💁‍♀️ >_< 😒 ;-) 💃	79	13%

- Follow-up Emoji poll currently running due to new Emojis, more participants expected with digital poll
  - Already more than 1,150 participants
  - Sentiment again very homogeneously perceived

Please help us to uncover the mystery:

emoji-poll.de/en

Key Take-away Messages

- **NFRs** are a **really interesting** field in research and in practice ☺
- There are **best practices** around that can help in handling them that **can also work for agile** settings
- **Crowd-RE** with text- and usage mining is a **great means** to support NFR handling
  - Most statements relate to qualities
- We see the **combination of text- and usage- mining** as an underrepresented, but very promising area for our RE community
- We should keep an open eye for **lightweight** approaches