Exam Date

Written exam

Onsite @ Garching

18th of August, 8:45 am

Duration: 60 min

Requirements Engineering

Lecture 6

Prof. Dr. Alexander Pretschner

Chair of Software & Systems Engineering
TUM Department of Informatics
Technical University of Munich

Orientation

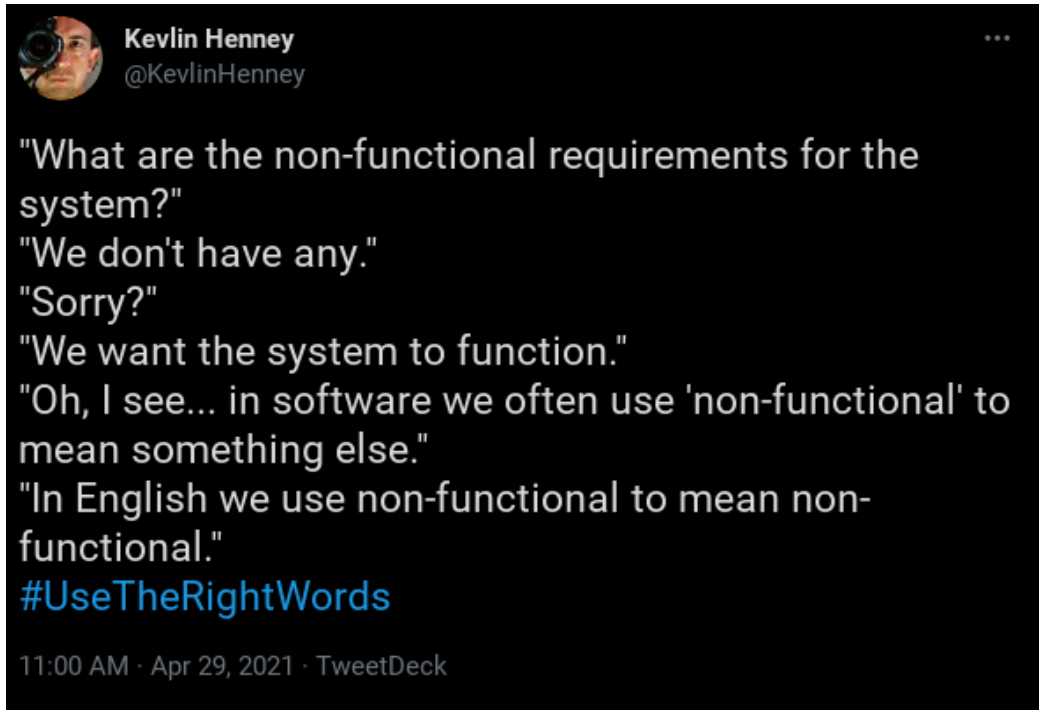
Recap:

- Goal Model
- GORE

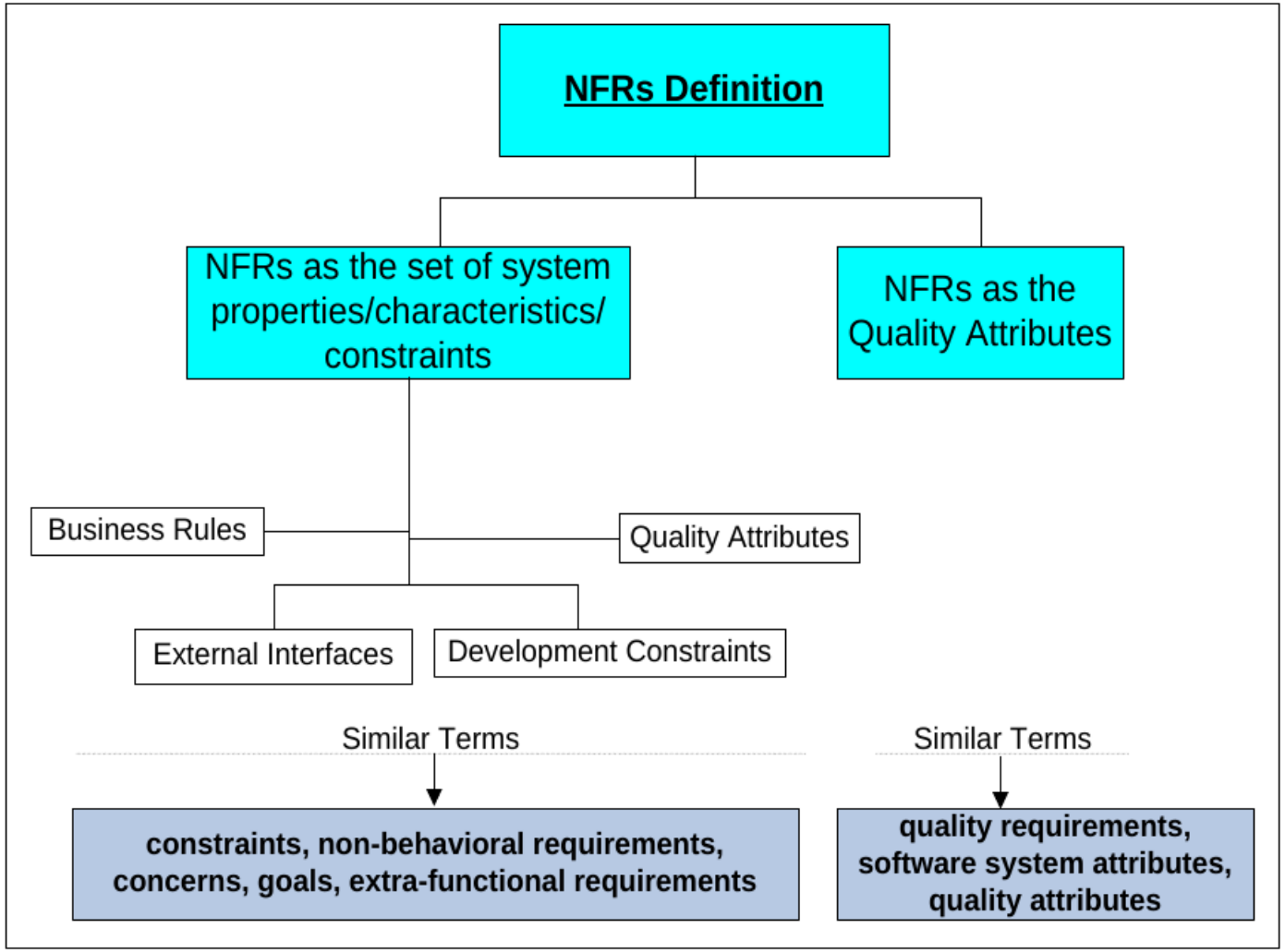
Coming up:

- Non-Functional Requirements
- Ethics in the RE process

Defining Non-Functional Requirements



https://twitter.com/KevlinHenney/status/1387693150785966083?s=2

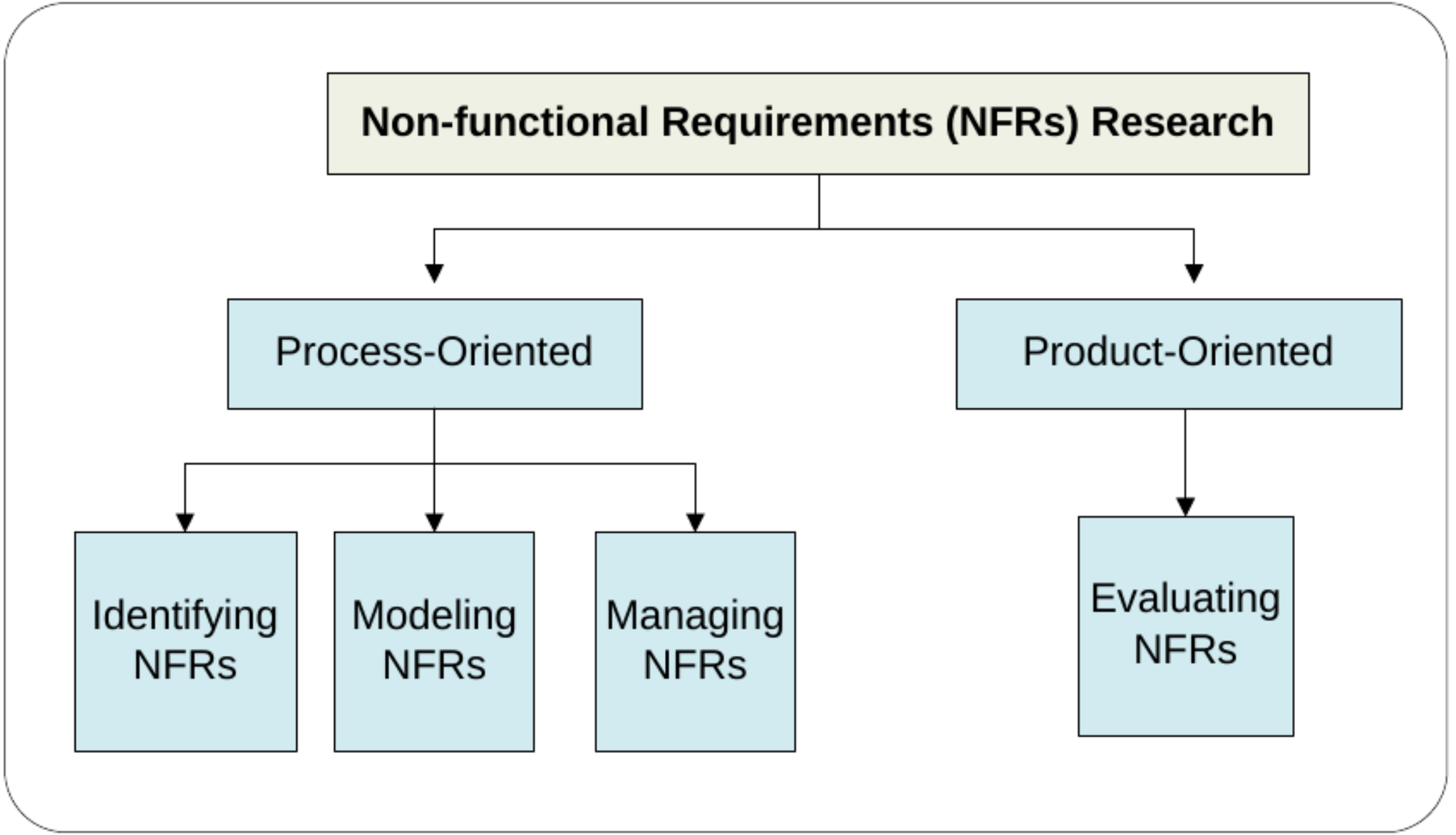


Mairiza et al. (2010)

Table 1. Definitions of the term *'non-functional requirement(s)'* (listed in alphabetical order of sources)

Source	Definition
Antón [1]	Describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate.
Davis [3]	The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability.
IEEE 610.12 [6]	Term is not defined. The standard distinguishes design requirements, implementation requirements, interface requirements, performance requirements, and physical requirements.
IEEE 830-1998 [7]	Term is not defined. The standard defines the categories functionality, external interfaces, performance, attributes (portability, security, etc.), and design constraints. Project requirements (such as schedule, cost, or development requirements) are explicitly excluded.
Jacobson, Booch and Rumbaugh [10]	A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement.
Kotonya and Sommerville [11]	Requirements which are not specifically concerned with the functionality of a system. They place restrictions on the product being developed and the development process, and they specify external constraints that the product must meet.
Mylopoulos, Chung and Nixon [16]	"... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) There is not a formal definition or a complete list of nonfunctional requirements."
Ncube [17]	The behavioral properties that the specified functions must have, such as performance, usability.
Robertson and Robertson [18]	A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property.
SCREEN Glossary [19]	A requirement on a service that does not have a bearing on its functionality, but describes attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.
Wiegers [21]	A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior.
Wikipedia: Non-Functional Requirements [22]	Requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors.
Wikipedia: Requirements Analysis [23]	Requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).

Glinz (2007)


Non-functional Requirements (NFRs) Research
Process-Oriented
Product-Oriented
Identifying NFRs
Modeling NFRs
Managing NFRs
Evaluating NFRs

Examples

1. Accessibility/Access Control
2. Accountability
3. Accuracy
4. Adaptability
5. Additivity
6. Adjustability
7. Affordability
8. Agility
9. Analyzability
10. Anonymity
11. Atomicity
12. Attractiveness
13. Auditability
14. Augmentability
15. Availability
16. Certainty
17. Changeability
18. Communicativeness
19. Compatibility
20. Completeness
21. Complexity/Interacting Complexity
22. Composability
23. Comprehensibility
24. Comprehensiveness
25. Conciseness
26. Confidentiality
27. Configurability
28. Conformance
29. Consistency
30. Controllability
31. Correctness
32. Customizability
33. Debuggability
34. Decomposability
35. Defensibility
36. Demonstrability
37. Dependability
38. Distributivity
39. Durability
40. Effectiveness
41. Efficiency/Device Efficiency
42. Enhanceability
43. Evolvability
44. Expandability
45. Expressiveness
46. Extendability
47. Extensibility
48. Fault/Failure Tolerance
49. Feasibility
50. Flexibility
51. Formality
52. Functionality
53. Generality
54. Immunity
55. Installability
56. Integratability
57. Integrity
58. Interoperability
59. Learnability
60. Legibility
61. Likeability
62. Localizability
63. Maintainability
64. Manageability
65. Maturity
66. Measurability
67. Mobility
68. Modifiability
69. Nomadicity
70. Observability
71. Operability
72. Performance/Efficiency/ Time or Space Bounds
73. Portability
74. Predictability
75. Privacy
76. Provability
77. Quality of Service
78. Readability
79. Reconfigurability
80. Recoverability
81. Reliability
82. Repeatability
83. Replaceability
84. Replicability
85. Reusability
86. Robustness
87. Safety
88. Scalability
89. Security/Control and Security
90. Self-Descriptiveness
91. Simplicity
92. Stability
93. Standardizability/ Standardization/Standard
94. Structuredness
95. Suitability
96. Supportability
97. Survivability
98. Susceptibility
99. Sustainability
100. Tailorability
101. Testability
102. Traceability
103. Trainability
104. Transferability
105. Trustability
106. Understandability
107. Uniformity
108. Usability
109. Variability
110. Verifiability
111. Versatility
112. Viability
113. Visibility
114. Wrappability

Mairiza et al. (2010)

Fuzzy

have definition and attributes

accessibility; adaptability; availability; efficiency; fault tolerance; functionality; integratability; integrity; maintainability; modifiability; performance, portability; privacy; readability; reliability; reusability; robustness; safety; scalability; security; testability; understandability; and usability

have definition

accuracy; analyzability; attractiveness; changeability; communicativeness; completeness; complexity; composability; confidentiality; consistency; correctness; defensibility; dependability; evolvability; extendability; flexibility; immunity; installability; interoperability; learnability; likeability; localizability; maturity; operability; quality of service; recoverability; replaceability; stability; suitability; survivability

without definition and attributes

accountability; additivity; adjustability; affordability; agility; anonymity; atomicity; auditability; augmentability; certainty; compatibility; comprehensibility; comprehensiveness; conciseness; configurability; conformance; controlability; customizability; debuggability; decomposability; demonstrability; distributivity; durability; effectiveness; enhanceability; expandability; expressiveness; extensibility; feasibility; formality; generality; legibility; manageability; measurability; mobility; nomadicity; observability; predictability; provability; reconfigurability; repeatability; replicability; self-descriptiveness; simplicity; standardizability; structuredness; supportability; susceptibility; sustainability; tailorability; traceability; trainability; transferability; trustability; uniformity; variability; verifiability; versatility; viability; visibility; wrappability

Mairiza et al. (2010)

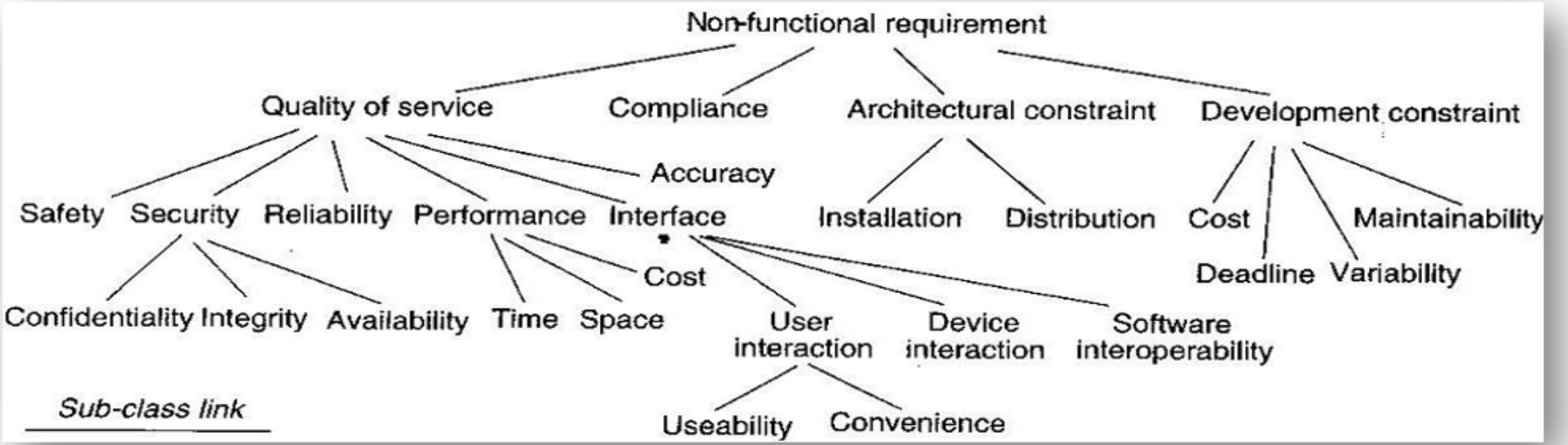
Examples

NFRs	Definition	Attributes
Performance	requirements that specify the capability of software product to provide appropriate performance relative to the amount of resources needed to perform full functionality under stated conditions	response time, space, capacity, latency, throughput, computation, execution speed, transit delay, workload, resource utilization, memory usage, accuracy, efficiency compliance, modes, delay, miss rates, data loss, concurrent transaction processing
Reliability	requirements that specify the capability of software product to operates without failure and maintains a specified level of performance when used under specified normal conditions during a given time period	completeness, accuracy, consistency, availability, integrity, correctness, maturity, fault tolerance, recoverability, reliability, compliance, failure rate/critical failure
Usability	requirements that specify the end-user-interactions with the system and the effort required to learn, operate, prepare input, and interpret the output of the system	learnability, understandability, operability, attractiveness, usability compliance, ease of use, human engineering, user friendliness, memorability, efficiency, user productivity, usefulness, likeability, user reaction time
Security	requirements that concern about preventing unauthorized access to the system, programs, and data	confidentiality, integrity, availability, access control, authentication
Maintainability	requirements that describe the capability of the software product to be modified that may include correcting a defect or make an improvement or change in the software	testability, understandability, modifiability, analyzability, changeability, stability, maintainability compliance

Mairiza et al. (2010)

NFRs in Application Domains

Application Domain	Relevant NFRs
Banking and Finance	accuracy, confidentiality, performance, security, usability
Education	interoperability, performance, reliability, scalability, security, usability
Energy Resources	availability, performance, reliability, safety, usability
Government and Military	accuracy, confidentiality, performance, privacy, provability, reusability, security, standardizability, usability, verifiability, viability
Insurance	accuracy, confidentiality, integrity, interoperability, security, usability
Medical/Health Care	communicativeness, confidentiality, integrity, performance, privacy, reliability, safety, security, traceability, usability
Telecommuni-cation Services	compatibility, conformance, dependability, installability, maintainability, performance, portability, reliability, usability
Transportation	accuracy, availability, compatibility, completeness, confidentiality, dependability, integrity, performance, safety, security, verifiability

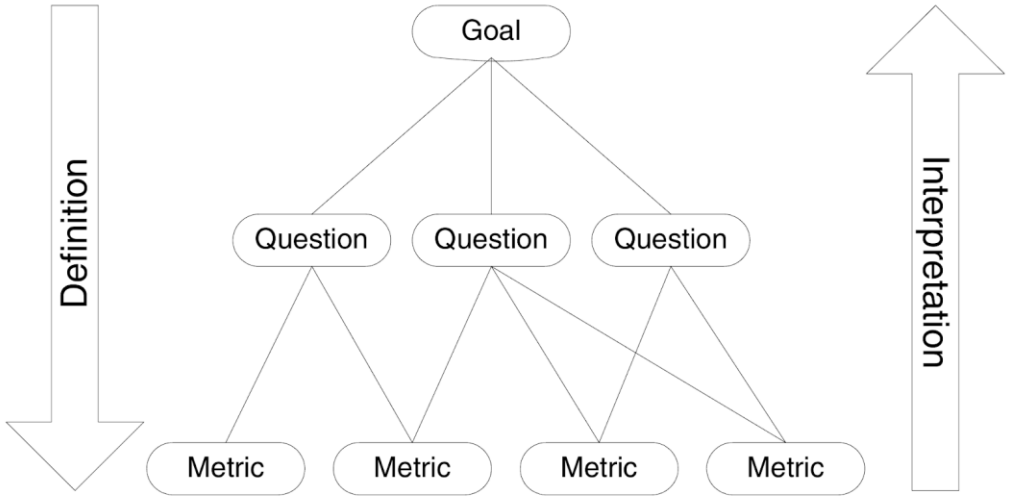


Van Lamsweerde (2009)

How can we measure NFRs?

Remember: Goal Question Metric (Lecture 5)
We define NFRs as goals and can then use GORE frameworks to model them and their dependencies.
It is important to define the metrics as early as possible.

⇨ This makes goals precise.



Goal	Purpose Issue Object Viewpoint Context	Improve the reliability of product X for the user within organisation Y
Question Metric	Q1 M1	What is the probability of failure on demand for function Z? POFOD
Question Metric Metric	Q2 M2 M3	How erroneous does function Z behave? MTTF MTTR
...	...	...

Example

“Keep Scooters Operational”

➔ Hard to give a formal definition.

We can provide an empirical quality metric:

➔ “At any point in time, 95% of scooter should be operational.”

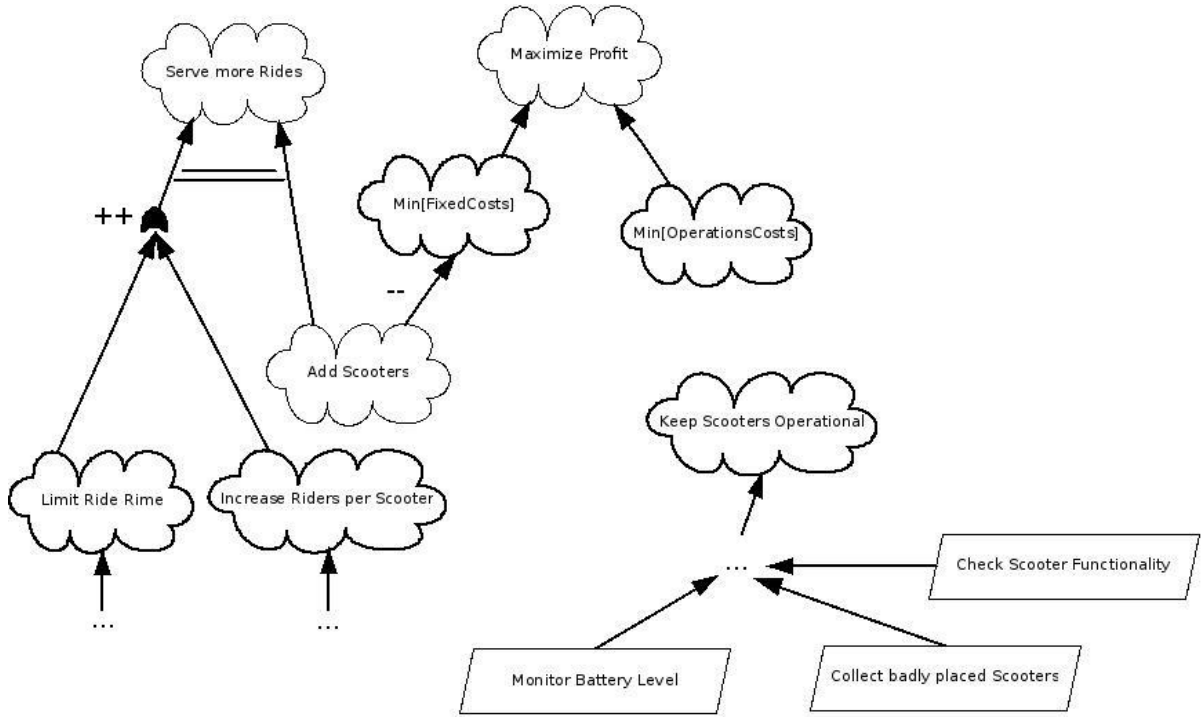
But: what does “operational” mean?

Translating NFRs to functional requirements

A scooter is “operational” when:

- Its battery is charged above a minimum level.
  ➔ Battery level > 30%
- The brakes discs are stronger than a minimum level
  ➔ Brake disc strength > 10mm.
- It is parked at a collection point.
  ➔ GPS location within geofence.

⇨ This means that often **non-functional** requirements are **sources** for **functional** requirements.



Functional vs. Non-functional

Very hard to differentiate!

➔ Example: "The airbag must be inflated in 0.25 sec"

⇨ This describes the **behavior** and the **performance characteristics**.

⇨ An airbag that inflates in 5 minutes is not an airbag, but a balloon.

How can we deal with **ethical** requirements?

- So far, we covered NFRs where “more is better”.
  - Security, Performance, Usability,...

- How can we deal with even more ambiguous requirements
  - Do no harm.
  - Improve humanity.
  - Be Fair.

- Important:
  - Algorithmic bias (e.g., not showing high paying job to women)
  - Undesirable outcomes (Uber is increasing traffic in cities, AirBnB increases rent in city centers)
  - Cambridge Analythica (Facebook data used to manipulate voters)

Ethical Deliberation in Software Engineering

Code of Conducts?

Ethics is the study of the right (and wrong) course of action.
Code of Conducts are normative statements that describe how an individual ought to act.
There exists a myriad of different code of conducts that are all over the place (ACM, IEEE, GfI, companies like Google, Facebook, Amazon, etc.)
It is unclear if they work and to what degree they actually influence people's behavior.
The content of the codes is bewildering and the recommendations are mostly either trivial, confusing, unstructured or simply empty.
I believe this is necessarily the case: software is context-specific, so why shouldn't ethical concerns be as well?

McNamara et al. (2018) find no evidence that CoCs influence behavior
Are CoCs actually useful for the software engineer?

Values and maxims all around...

- Privacy, Transparency, Fairness, Accountability, Well-being, Responsibility, Interoperability, Dignity, Participation, Data Protection, Safety & Security, Non-Discrimination, Autonomy, Inclusion, Equality, Harm avoidance....
- AI software and hardware systems need to be human-centric. EU Guidelines on Ethics in Artificial Intelligence 2019
- Be honest and trustworthy. ACM 2018
- Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks. ACM 2018
- Awareness of Misuse–A/IS creators shall guard against all potential misuses and risks of A/IS in operation. IEEE 2019
- Engineers shall at all times strive to serve the public interest. NSPE 2019

When values collide...

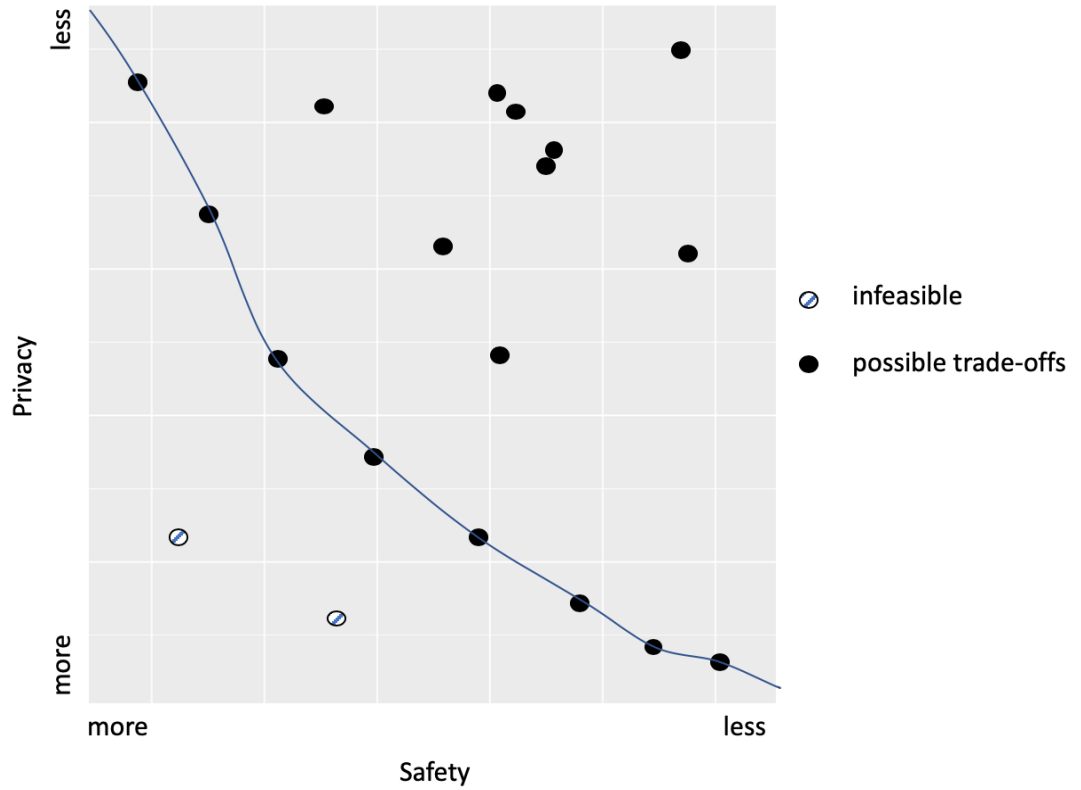
CoCs do not tell us what to do when values are mutually exclusive (which happens ALL the time) like privacy vs. transparency, freedom vs. safety, etc.

When we try to optimize a model/software that relies on two (or more) values all we know is that it should satisfy Pareto Optimality.

Only solutions on the frontier are reasonable

But the all important question remains:

Where on the curve?



Ethical deliberation in systems engineering

Mostly, management will provide the outline of what the product should be - in such scenarios if they do not share the company's vision, they can only quit.
Major ethical problems need to be considered at the societal or organizational levels.
For example: The high-level goal is to **"Find a way to keep birds off some property."**
The question whether we may restrict free movement of birds etc. is not to be answered by the engineer!
The degree of freedom in designing could be for example:

- A team member with a background in construction might suggest to use spikes on rims and poles where birds like to sit.
- One with a background as a falconer might suggest to get a falcon to scare off other birds.
- A sound engineer might suggest using high pitched noises.
- An environmental activists will catch the birds and find for them another place to live.

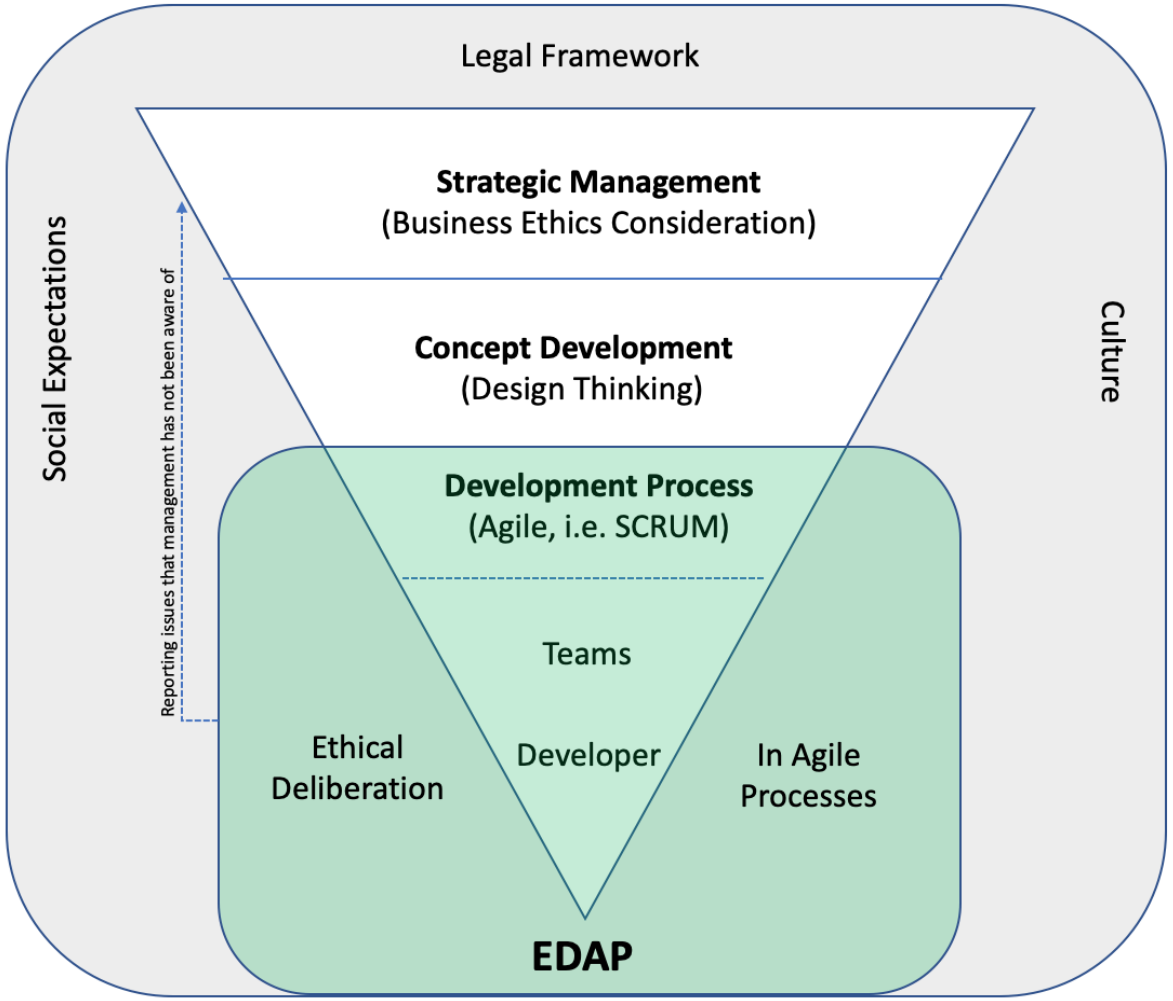
The place and the limits of ethical consideration by software engineers

- Software producers are under increasing pressure to design “ethical” software
- Often, this is followed by the claim that software engineers should consider all sorts of ethical issues while designing their products
- These demands, however, often exceed the capabilities or the power of software engineers
- **Therefore, it is important to define the domain, the scope and the limits of ethical considerations that can and should be performed by software engineers and their respective teams**
- The domain of influence of software developers to implement ethical design is within given parameters that have already been set by laws, culture and business decisions

Locating the influence of the individual

Many decisions are taken by management (e.g., build weapons) or at the design level (e.g., use gamification).
Developers only have a limited room to maneuver.

Here they can, for example, decide to make a website accessible or avoid storing some specific data.



How to address ethical issues?

1. **Descriptive Ethics:**
   Descriptive value mapping: Which values need to be addressed? What kind of trade-offs arise (privacy vs. transparency)?
2. **Normative Ethics:**
   Reflect upon values to highlight their relationships and importance for the technical object. In doing so, we aspire to find options and prioritize judgments.
3. **Applied Ethics/ Normative Design:**
   Think about how to transfer our normative thinking into design thinking.

Discussion: For the Scooter

How can we ensure all communities are served equally? And not just rich city centers?
How do we use the rider data? Sell it? Delete it? What if we need the extra revenue of selling the data?
How do we protect the environment from discarded scooters? Especially the toxic batteries?

Outline and Outlook

Terms and Definitions
Core activities
Quality models for Requirements
Engineering Models
Stakeholders and Elicitation Techniques
Goals and Goal-oriented RE

Non-functional requirements

Functional requirements
Formalization
Agile Processes
Requirements Management and Quality Assurance
Trends in Research

Further Reading

Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010, March). An investigation into the notion of non-functional requirements. In *Proceedings of the 2010 ACM Symposium on Applied Computing* (pp. 311-317).

Glinz, M. (2007, October). On non-functional requirements. In *15th IEEE International Requirements Engineering Conference (RE 2007)* (pp. 21-26). IEEE.

Gogoll, J., Zuber, N., Kacianka, S. *et al.* Ethics in the Software Development Process: from Codes of Conduct to Ethical Deliberation. *Philos. Technol.* (2021). https://doi.org/10.1007/s13347-021-00451-w

DeVries, B., & Cheng, B. (2019, September). Goal-based modeling and analysis of non-functional requirements. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)* (pp. 261-271). IEEE.

Brito, I., & Moreira, A. (2004). Integrating the NFR framework in a RE model. *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, 28.

Zuber, N., Kacianka, S., Nida-Rümelin, J. & Pretschner, A. (2020): Ethical deliberation for Agile software processes: EDAP manual. Hengstschläger, M (ed.): *Digital Transformation and Ethics*.