Jan Jürjens, Kurt Schneider (Hrsg.): Software Engineering 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 105

Are "Non-functional" Requirements really Non-functional?
An Investigation of Non-functional Requirements in Practice

Jonas Eckhardt[1], Andreas Vogelsang[2], Daniel Méndez Fernández[1]

Abstract: Non-functional requirements (NFRs) are commonly distinguished from functional requirements (FRs) by differentiating *how* the system shall do something in contrast to *what* the system shall do. This distinction is not only prevalent in research, but also influences how requirements are handled in practice. NFRs are usually documented separately from FRs, without quantitative measures, and with relatively vague descriptions. As a result, they remain difficult to analyze and test. Several authors argue, however, that many so-called NFRs actually describe behavioral properties and may be treated the same way as FRs. In this paper, we empirically investigate this point of view and aim to increase our understanding on the nature of NFRs addressing system properties. Our results suggest that most "non-functional" requirements are *not* non-functional as they describe behavior of a system. Consequently, we argue that many so-called NFRs can be handled similarly to FRs.

Keywords: Non-functional requirements, classification, model-based, empirical studies

1 Summary

Although the importance of NFRs for software and systems development is widely accepted, the discourse about how to handle NFRs is still dominated by how to differentiate them exactly from FRs [Br16, Gl07]. One point of view is that the distinction is an artificial one and we should rather differentiate between *behavior* (e.g., response times) and *representation* (e.g., programming languages). The underlying argument is that most NFRs actually describe behavioral properties [Gl07] and should be treated the same way as FRs in the software development process [Br16]. Behavioral properties subsume classical FRs, such as *"the user must be able to remove articles from the shopping basket"* as well as NFRs which describe behavior such as *"the system must react on every input within 10ms"*. Representational properties include NFRs that determine how a system shall be syntactically or technically represented, such as *"the software must be implemented in the programming language Java"* [Br16]. In this paper, we empirically investigate this point of view and aim to increase our understanding on the nature of NFRs addressing system properties. To this end, we classify 530 NFRs extracted from 11 industrial requirements specifications with respect to their kind. Our results show that 75% of the requirements labeled as "non-functional" in the considered industrial specifications describe system behavior and only 25% describe the representation of the system. As behavior has many facets, we further classify behavioral NFRs according to the *system view* they address (interface, architecture, or state), and the *behavior theory* used to express them (syntactic,

[1] Technische Universität München, Institut für Informatik, {eckharjo,mendezfe}@in.tum.de
[2] Technische Universität Berlin, DCAITI, andreas.vogelsang@tu-berlin.de

logical, probabilistic, or timed) [Br16]. Based on this fine-grained classification, we discuss the implications we see on handling NFRs in the software engineering disciplines, e.g., testing or design. The full paper can be found in [EVMF16].

2 Results & Conclusion in a Nutshell

Fig. 1 shows how many NFRs describe system behavior and Fig. 2 shows which *system views* behavioral NFRs address. The tables show the distribution of behavioral and representational NFRs (Fig. 1) and the distribution of NFRs with respect to the *system view* they address (Fig. 2). The bar charts show these distributions with respect to the quality characteristics of the ISO 9126.

Behavioral vs. Representational	**count**	**%**
Behavioral	396	74.7%
– Black-box	273	51.5%
– Glass-box	123	23.2%
Representational	134	25.3%

System view	**count**	**%**
Interface	273	68.9%
Architecture	85	21.5%
State	38	9.6%

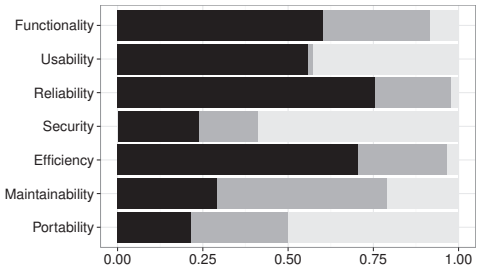

Fig. 1: *Black-box* (black), *glass-box* (dark gray), and *representational* (light gray).

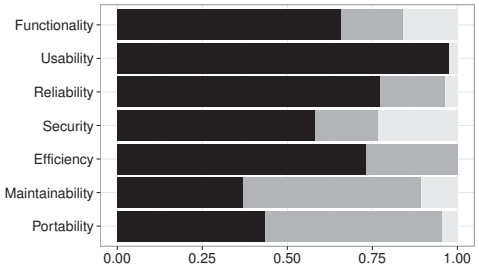

Fig. 2: *Interface* (black), *architecture* (dark gray), and *state* (light gray).

Based on our results, we furthermore discuss that FRs were often labeled as NFRs, NFRs are often specified by reference to standards, and that only few NFRs deal with architectural aspects. Finally, we conclude that most "non-functional" requirements are misleadingly declared as such because they actually describe behavior of the system. This in turn means that many so-called NFRs can be handled similarly to FRs.

References

[Br16] Broy, M.: Rethinking Nonfunctional Software Requirements: A Novel Approach Categorizing System and Software Requirements. In: Software Technology: 10 Years of Innovation in IEEE Computer. 2016.

[EVMF16] Eckhardt, J.; Vogelsang, A.; Méndez Fernández, D.: Are "non-functional" requirements really non-functional? In: ICSE. 2016.

[Gl07] Glinz, M.: On non-functional requirements. In: RE. 2007.