Non-functional and functional requirements: are they equally relevant?

Angelo Gargantini

University of Bergamo, Italy

Are they equally relevant?

- ignoring (or underestimate) them can determine the failure of a software project
  - Yes
- can cause a failure in an application
  - Yes
- should be dealt with from the beginning of software development
  - yes
- are they treated equally:
  - No:
    - We have methods to formalize, validate and test functional requirements
    - Not so for non functional requirements

NFR as second class citizens

- Why?
  - Representation: no clear semantics for NFRs
    - Sometimes they are non functional only because there are not defined in details
  - Methodology:
    - NFRs refer more to the process than to the product ?
  - Or …

Promotion of NFRs

- Partially because of lack of formalization
- Indeed, sometimes they can be “promoted”
- Example:
  - The system shall ensure that data is protected from unauthorised access → non functional
  - The system shall include a user authorisation procedure where users must identify themselves using a login name and password. Only users who are authorised in this way may access the system data → functional
- NFR only because we have not formalized them?
  - Sort of alibi if the sw/prj fails?
- Not equally formalized → not equally considered
  - Why we are so unwilling to formalize them?

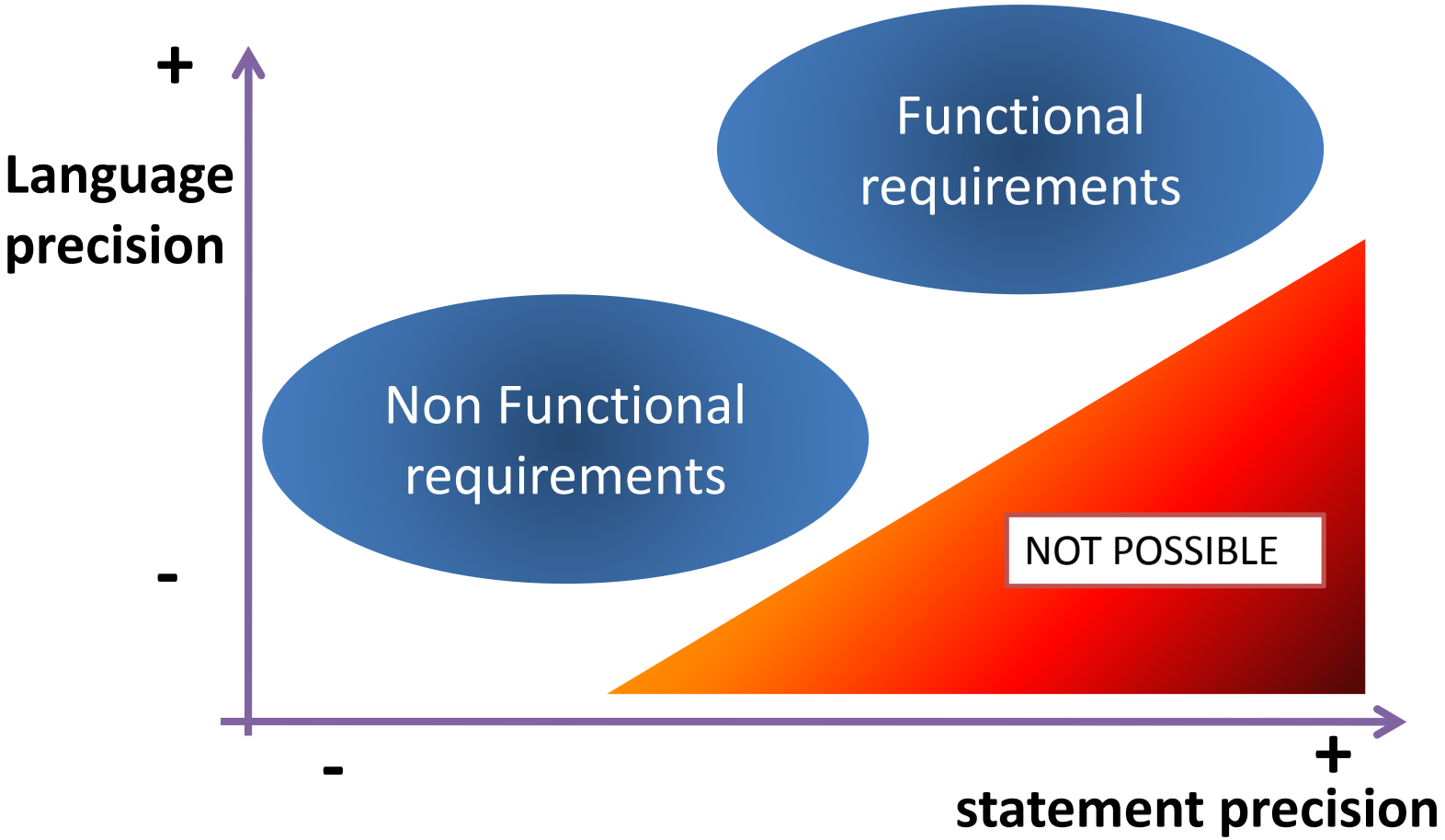
Why not equally formalized

- Is there a need of a clear semantics ?
  - Yes, if they want to be equal!
- Are they equally precise ?
  - No
  - Example [Mylopoulous et al. TSE 92]:
    - software visibility: “there shall be no more than X branches per 1,000 lines of code”
    - The requirement is not accomplished if X+1? And X-1 is fine?

Why no clear semantics of NFRs?

- A **formal** language is normally used to express **precise** statements
  - assert x = 100
- A rigorous language can still be used to make imprecise statement
  - assert x > 98 and x < 102
- The converse is not true: an imprecise language cannot express precise statements
  - A non rigorous language can express only imprecise statements
    - X is about 100
- Most FRs are precise → require a formal language → enable formal automatic analysis
- No need to define a rigorous language for imprecise statement

No need of precision for NFR?



Harel and Rumpe, Meaningful modeling: the semantics of “semantics”

How should be a representation of NFRs?

- No reason to be as strict as for functional requirements
  - Would you use first order logic the represent *usability*?
- A methodology for NFRs must be **qualitative**
  - Chung, Nixon, Yu, Mylopoulos @ toronto NFR framework in 1999

Qualitative reasoning

- Qualitative reasoning
  - "We won't say that a NFR is accomplished or not"
  - "Only that design decisions contribute positively or negatively towards a particular goal"
- "Lightweight semi formal methods"
  - graphical framework
  - Is this enough to efficiently reason about NFRs?
- NOT?

NFRs → NFNR

- If the NFR framework is only a guideline
- If the reasoning is only qualitative
  - No guarantee , only heuristics

- No way to say if [Jackson&Zave]

  W, S |- R

  No real requirements
  - NON FUNCTIONAL NON REQUIREMENTS

The end of NFRs

- The NFRs are destined to be incorporated by process/methodology – only as design rationale
  - To keep only a link between a NFR and design decisions
- NFRs would be granted by design
  - example:
    - Use this design pattern and you get XXX
    - Use this architecture and you get YYY
    - Use XP and you get ZZZ
- no more need of explicit formal representation, validation, testing for NFRs ?
- Only reasoning about the designs