On the management of non-functional requirements

Dr. Tullio Vardanega
European Space Research and Technology Centre
and University of Padua

TU Delft, 12 November 2002

Outline of the talk

- What are nonfunctional requirements (NFR)
- How do they differ from functional requirements
- Where do they originate from and when
- How do NFR impact the development process
  - What is the problem with them
  - Definition, engineering and verification issues
  - Strategies for incorporation
  - Integration versus separation

What are NFR

- NFR concern such characteristics as:
  - Safety
  - Reliability
  - Timeliness (predictability)
  - Reusability
  - Portability, etc.
- Functional requirements tell what the system has to do
- NFR tell how the system functions have to be achieved

What are NFR (cont’d)

- NFR originate at system and at software level, for example:
  - Safety and reliability are system-level NFR
  - Reusability and portability are software-level NFR
  - Predictability pertains to both levels
- NFR concern product and process aspects

Example 1

Implementation techniques may facilitate verification

6. Implementation Techniques
   6.1 Rationale
   6.2 Coding Rules

Timing requirements (e.g. deadline) specify, in the time domain, what functional requirements specify in the value domain

For a complex software system to perform as expected

What are NFR (cont’d)

- Project requirements include and exceed product requirements
  - Both sets originate from multiple sources with often independent and differing perspectives

Developing organisation

Ordering customers

Product requirements
NFR impact

- A large proportion of software projects in virtually all application domains often fail in several respects, e.g.:
  - Late shipment, budget overrun
  - Underutilisation in operation
  - Laborious maintenance, scarce portability

NFR impact (cont’d)

- The root of the failure can often be found in mismanaged NFR, e.g.:
  - Lack of specification or misinterpretation
  - Incomplete or erroneous specification
  - Insufficient verification
  - These deficiencies are less apparent than the violation of functional requirements

NFR impact (cont’d)

- NFR are more difficult to express than functional requirements
- NFR arise predominantly at project level
- Product requirements are typically functional and often arise earlier than NFR
- This independence of origin often makes them conflict in subtle ways

Example 2

- A whole catalogue of major software failures
- Lots of other business sectors could be added to the list

Example 3

- How can we translate a system-level reliability requirement expressed as a .9993 fphs (<1 failure in 60 days of operation) to software requirements
- How can we possibly verify it

Seminario: Trattamento dei requisiti non funzionali
NFR impact (cont’d)

- Specification (cont’d)
  ✓ NFR are often expressed in forms that are hard to quantitatively or objectively verify
  ✓ their verification is frequently qualitative and subjective
  ✓ we lack standardised and well-understood techniques to reduce subjectivity
  ✓ we have little support for the capture of NFR and poor means to raise them to proper design and/or process drivers

Example 4

Failure mode effects and criticality analysis

- Specification
- No enforcement means
- Requirements for design and coding standards
- Verification
- Manual verification
- Project time
- Retrofit on implementation

NFR impact (cont’d)

- Engineering
  ✓ Typical challenges:
    - how can we align design methodology and design patterns to preferred/prescribed coding standards
    - how can we avoid destructive implementation of fault tolerance provisions against late understanding of needs

NFR impact (cont’d)

- Engineering (cont’d)
  ✓ Typical challenges:
    - How can we ensure that all concerned disciplines pull the project in one and the same direction
    - How can we strike a good balance between efficiency, verifiability and reuse

Example 5

Integrated Modular Avionics

- Application #1
- Application Executive interface
- Module Operating System
- Module Operating System
- Partition Operating System

An avionics standard like IMA will considerably restrict the design and implementation freedom, while allowing multi-threaded, multi-language, distributed applications.
NFR impact (cont'd)
- Verification
  ✓ NFR are difficult to verify (at process level) and to validate (at product level)
  ✓ Difficulties are:
    • Technical ('how to')
    • Technological ('by what means')
    • Programmatic ('when and to what extent')

NFR impact (cont’d)
- Verification (cont’d)
  ✓ Definitions
    • Verification: confirmation, by examination or provision of objective evidence, that specified requirements have been fulfilled
    • Verification concerns the process
    • Validation: confirmation [...] that the product meets the specified requirements
    • Validation concerns the product

Nominal process
- Aims at the implementation of functional requirements
  ✓ Includes processes specific to:
    • The developing organisation
    • The individual project within the organisation
  ✓ Project-level processes include:
    • Management processes
    • Technical processes (development) encompassing:
      - Engineering processes
      - Verification processes
    ✓ May all be carried out in full parallelism

Nominal process (cont’d)
- The purpose and order of execution of the nominal processes is determined by multiple factors, e.g.:
  ✓ Application-domain specific constraints
  ✓ Technical and organisational considerations
  ✓ These factors may change in weight and nature during the life cycle, e.g.:
    ✓ Required team profile versus availability of personnel

Nominal process (cont’d)
- Irrespective of the extent of internal parallelism, the system life cycle proceeds across a set of stages
  ✓ Each stage has a distinct purpose and contribution
  ✓ The achievement of a stage represents a project milestone
    • This is a crucial measure of progress
**Nominal process (cont'd)**

- System development
- Subsystem development
- Hardware development
- Software development
- Human procedures

- Concept development
- Operation development
- Validation
- Design
- Integration & test
- Coding

**NFR management strategies**

- No standard definition of software NFR
  - Also referred to as 'characteristics'
  - Two broad classes:
    - Explicit
      - Typically defined by users and customers
      - Also derived from system-level requirements
      - Influence the ultimate acceptance of the product
    - Implicit
      - Reflect specific developers' concerns
      - Need dedicated processes to warrant satisfaction

**NFR management strategies (cont'd)**

- Explicit characteristics
  - Should be fully defined at the 'concept' stage of the software development
    - Use of int'l standard checklists may help the capture
  - Implicit characteristics
    - May concern the software product as a whole, e.g.:
      - Completeness, verifiability, compatibility
      - May concern constructive aspects of the product (technology, architecture), e.g.:
        - Timeliness (predictability), resource-efficiency
      - Should be defined during the 'specification / design' stages

**NFR management strategies (cont'd)**

- Can we use the same processes used for the implementation of functional requirements?
  - Should we perform them in parallel or in conjunction with the nominal processes
    - No general criteria prevent integration
    - Separation allows independence of verification responsibility
    - Integration is preferable (easier, cheaper) for most implicit characteristics
    - Separation is desirable for critical ones

**NFR management strategies (cont'd)**

- How to define adequate dedicated processes?
  - Use the Plan-Do-Check-Act model

**Example 6**

Parallel processes for the management of safety characteristics (explicit and crucial to acceptance)
**Example 7**

Integrated processes for the management of timeliness requirements
(implicit and inherent to technical processes)

<table>
<thead>
<tr>
<th>Software specification</th>
<th>Software design</th>
<th>Software coding</th>
<th>Software integration and test</th>
<th>Software verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time specification</td>
<td>Real-time design</td>
<td>Real-time coding</td>
<td>Real-time testing to support physical model</td>
<td>Real-time verification to prove physical model</td>
</tr>
<tr>
<td>Software logical model</td>
<td>Software physical model</td>
<td>Software physical model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Scheduling analysis
- Timing analysis
  (prediction, estimation, measurement)

**Conclusion**

- NFR will be increasingly more dominant in software-intensive systems
- There is no 'best' solution to their management
- Different solutions for different domains
- Integration works well for 'consolidated' NFR
- Separation sanctions the criticality of NFR

No longer a lonely ride across the desert

Not yet an easy strike