Non-Functional Requirements

Ana Moreira

FCT/UNL

Requirements elicitation

- Identify functional and technical system requirements
- Identify and understand the different types of users who will be involved in investigating system requirements
- Techniques: questionnaires, review of documentation, interviews, observation and shadowing, etc.

2

Types of requirements

- Functional requirements
  - Activities the system must perform
  - Based on procedures and business functions
  - Documented in analysis models
- Technical (non-functional) requirements
  - Describe operating environment and quality objectives
  - Documented in narrative descriptions of technical requirements

3

What are Non-Functional Requirements (NFRs)?

- NFRs are also known as **Quality Attributes** [Chung 93, Boehm 96]
- Unlike functional requirements, NFRs state **constraints to the system as well as particular behaviour** that the system must have
- NFRs are **always together with a functional requirement** [EAGLE 95, Chung 00]

4

Functional vs. NFRs

- Suppose we are dealing with a Clinical Analysis Laboratory software system
- There will be a functional requirement:
  - "The System must provide a data entry to input the results of tests to a patient's report"
- Associated to this requirement one may find the following Security NFR:
  - "Depending on the result of a test only the supervisor will be able to input this value to the patient's report"

5

Non-Functional X Functional requirements

- NFRs define global constraints:
  - on a software system or subsystem,
  - a functional requirement,
  - the development, or
  - deployment processes
- NFRs are often subjective in nature
- NFRs are generally stated
  - informally,
  - are often contradictory,
  - difficult to enforce during development,
  - difficult to evaluate for the customer prior to delivery

- FRs are local in nature (often defined as scenarios of interactions with the system)
- FRs are objective in nature
- FRs are generally stated
  - more formally (structured language, Use Cases/UML, formal methods),
  - usually do not conflict with each other,
  - achieving FRs can be enforced by todays design methods,
  - can be evaluated by customers

NFRs examples

Accessibility
Additivity
Accuracy
Accountability
Agility
Adjustability
Adaptability
Auditability
Affordability
Clarity
Capability
Capacity
Commonality
Code-space performance
Cohesuveness
Compatibility
Communication cost
Communication time
Component integration time
Completeness
Component integration cost

Conceptuality
Composability
Comprehensibility
Configurability
Conciseness
Confidentiality
Coordination cost
Consistency
Controllability
Cost
Coordination time
Correctness
Customer loyalty
Coupling
Customer evaluation time
Decomposability
Customizability
Data-space performance
Development cost
Degradation of service
Dependability
Diversity
Development time
Distributivity

Efficiency
Domain analysis cost
Domain analysis time
Evolvability
Elasticity
Enhanceability
External consistency
Execution cost
Extensibility
Flexibility
Fault-tolerance
Feasibility
Generality
Impact analyzability
Inspection cost
Inter-operability
Learnability
Maintenance cost
Mean performance
Modifiability
Nomadicity
Operability

Performance
Precision
Productivity
Promptness
Reconfigurability
Reengineering cost
Replaceability
Responsiveness
Risk cost
Safety
Security
Simplicity
Stability
Supportability
Testing time
Usability
Variability
Visibility
Verifiability
Validity
Wrappability
Versatility
.....

> 160 NFRs [Chung 00]

Why bother with NFRs? (1)

- Market demands more and more non-functional aspects to be implemented in information systems besides its functionality
- Recent works have shown that complex conceptual models must deal with non-functional aspects
  [Dardene 93, Mylopoulos 92, Chung 95]
- These non-functional requirements should be dealt within the process of NFR definition

8

A Recent Press Release

- Microsoft's chairman, Bill Gates, is steering his software empire onto a new strategic heading, putting **security** and **privacy** ***ahead of new capabilities*** [new functionality] in the company's products.
- In an e-mail to employees, Gates refers to the new philosophy as "**Trustworthy Computing**" and says his ***highest priority*** is to ensure that computer users continued to venture ***safely*** across an increasingly Internet-connected world.

Why bother with NFRs? (2)

- NFRs have a **strong influence on design decisions and implementation**:
  - They allow a wide spectrum of **possible solution choices**;
  - They **reveal other functional requirements** that would, otherwise, be ommitted and only found during late design and implementation phases.
- Netherless, there aren't many tools and techniques to handle them early enough during development.

10

Chung's NFR framework: what?

- It is process-oriented, providing techniques for justifying design decisions during the software development process
- It is goal-driven since it treats NFR as goals to be achieved
- It is different from traditional goal-oriented approaches [Dardenne 93, Anton 96]
  - As it uses the notion of *softgoa*l, which represents a goal that has no clear-cut criteria to determine whether they've been satisfied. (Satisficed)

11

NFR framework: main concepts

- Softgoal
- Interdependencies
  - Correlations, contributions and trade-offs
- Softgoal Interdependency Graph (SIG)
- Catalogue

[L.Chung, B.Nixon, E. Yu and J. Mylopoulos "Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers. ISBN 0-7923-8666-3. (2000)]

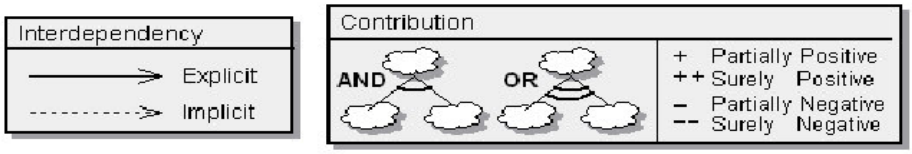
12

Sofgoal: basic unit to represent NFR

- Three kinds of softgoals:
  - NFR *softgoals* (or simply NFRs)
    - high-level non-functional requirements to be **satisficed**
  - *Operationalizing softgoals*
    - possible solutions (operations, processes, data representations) or design alternatives which help to achieve the NFR *softgoal*
  - *Claim softgoals*
    - justify the rationale and explain the context for a *softgoal* or interdependency link

Softgoal Interdependency Graph (SIG)

- Graph representing *softgoals* and their interdependencies



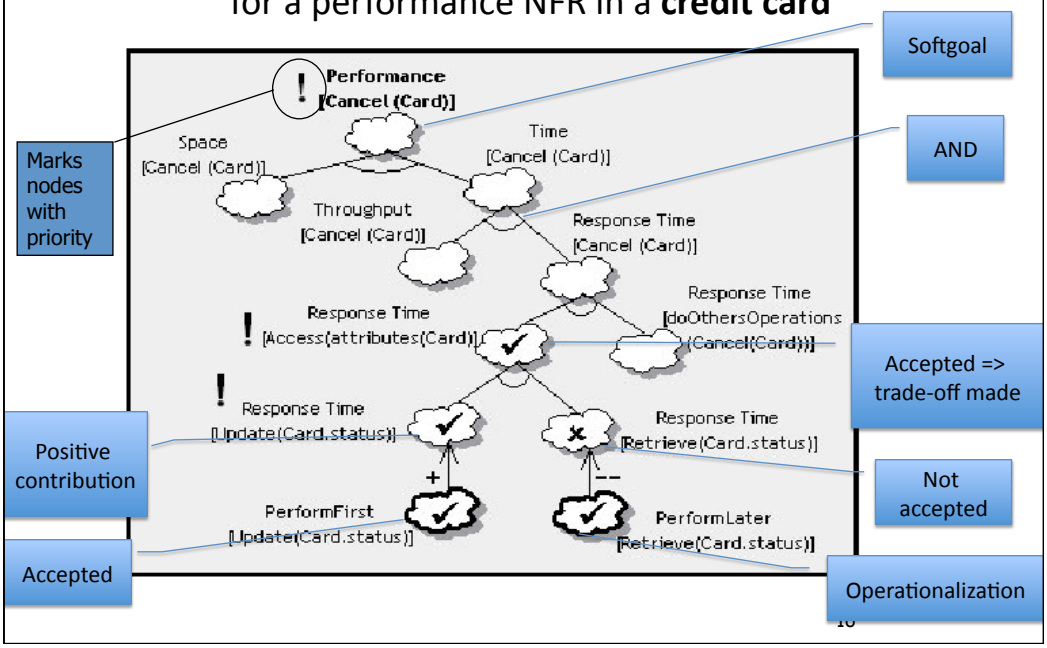
14

Catalogue

- Groups an organized body of design knowledge about NFRs
  - types
  - development techniques
  - correlations among *operationalizations* and NFR *softgoals*
- Can be used in different application domains
  - allowing composition of the SIG (Softgoal Interdependency Graph)

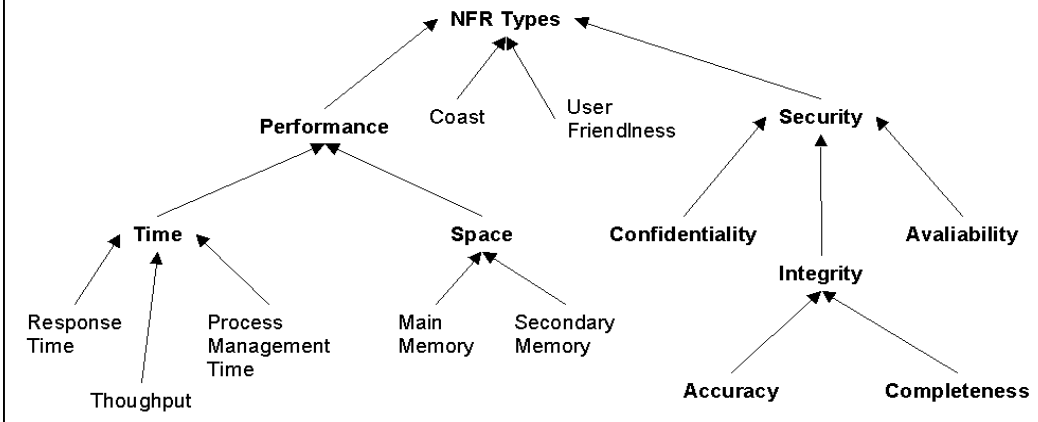
15

Catalogue: example for a performance NFR in a **credit card**



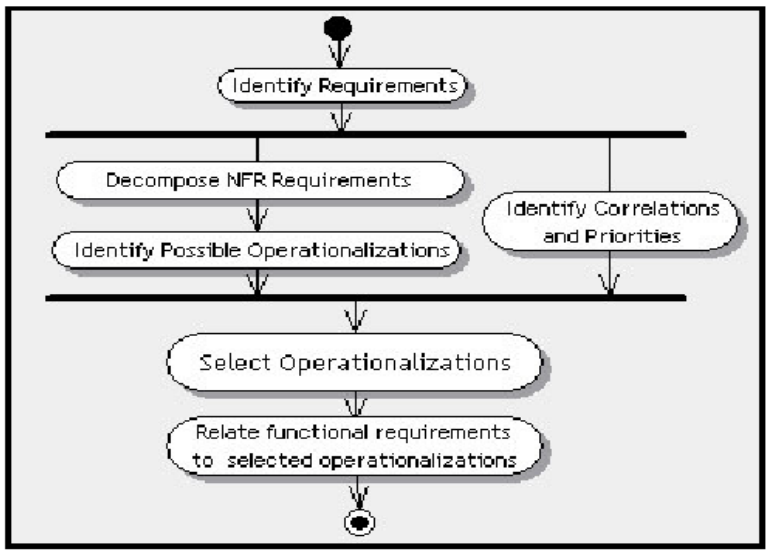
NFR type catalog

NFR Type Catalog



17

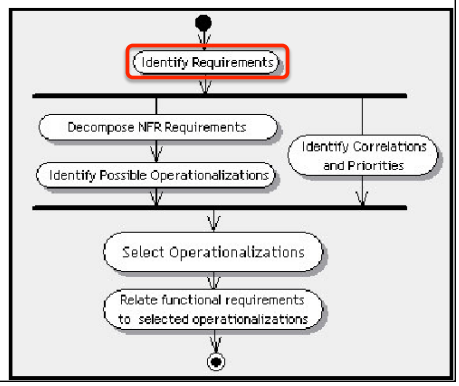
NFR process



18

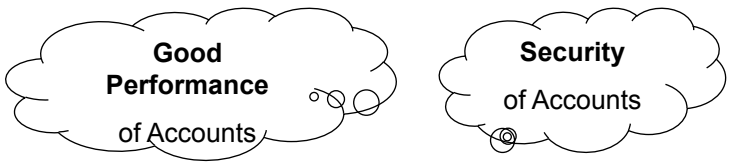
Identify NFRs

- List ambiguous, subjective, imprecise requirements – qualitative by nature that cannot be defined formally
- These are the Softgoals – it is not known how they will be implemented and what consequences they may provoke
- Initiate a SIG



Identify NFRs: example

"Credit cards control system"



- Two main softgoals
- Look up relevant information from existing catalogues

20

Catalogue *rationale*

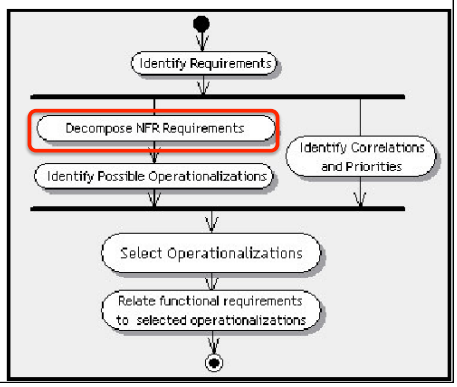
Should include:

- Design rational
  - Types of NFR
  - Methods (decomposition & operationalization)
  - Correlation (contribution) between softgoals
- Graphical representation for the developer choices

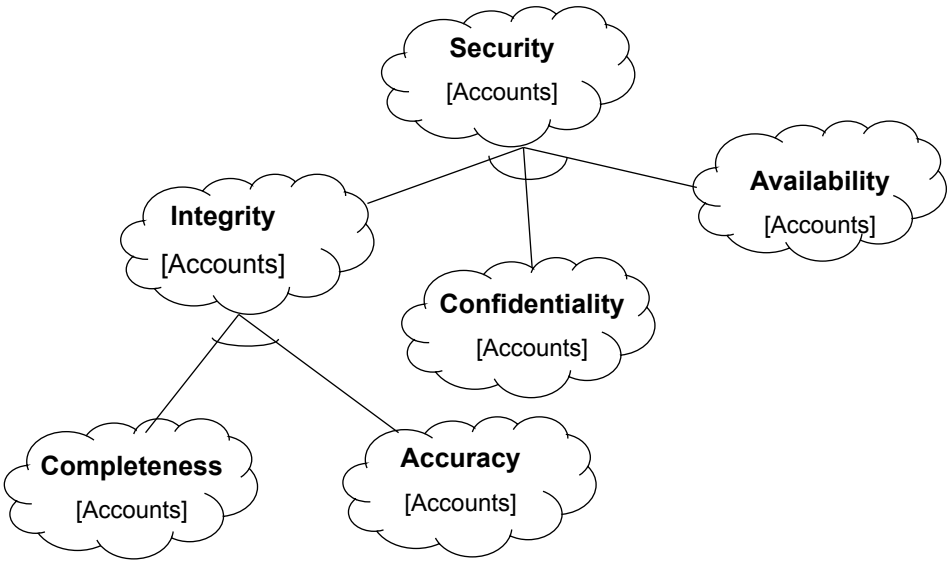
21

Decompose (refine) Softgoals

- AND relationship
  - means that if all offsprings are satisficed, then the parent will be satisficied
  - reduces the scope of the problem into more concrete sub-problems (transforms problems into subproblems)
- OR relationship
  - means that if any offspring is satisficed, then the parent will be satisficed
  - ilustrates alternative concepts (or notions)

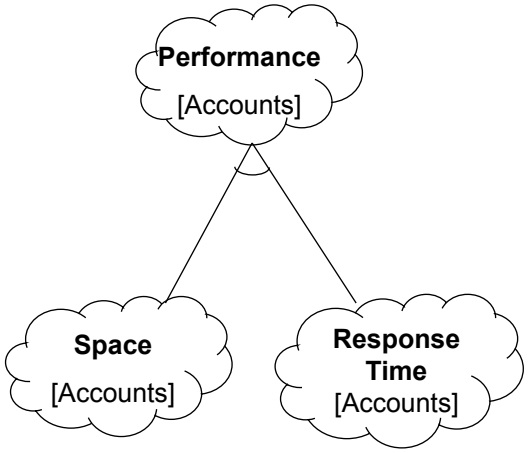


Decomposition: example (1)

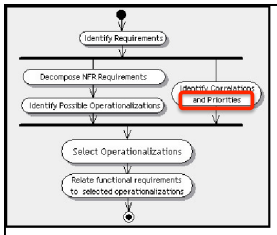


23

Decomposition: example (2)

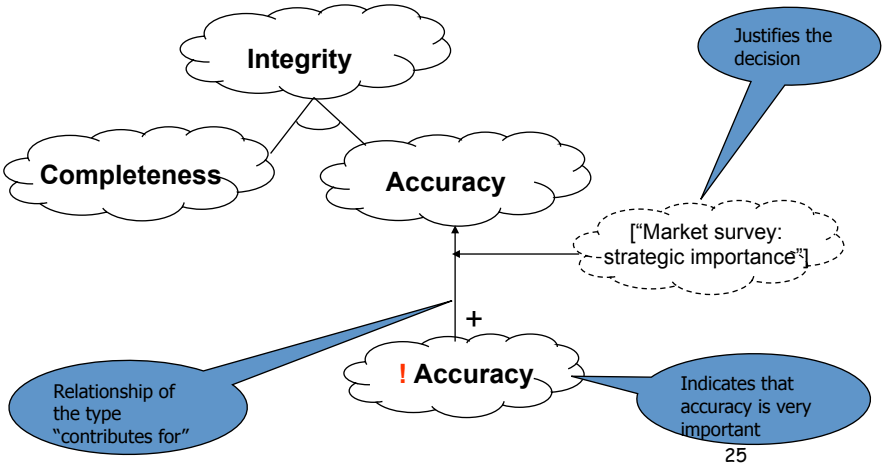


24

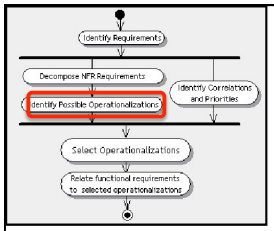


Identify Priorities

- SIG diagrams may be large and complex
- Assign priorities to important to keep "an eye" on important softgoals

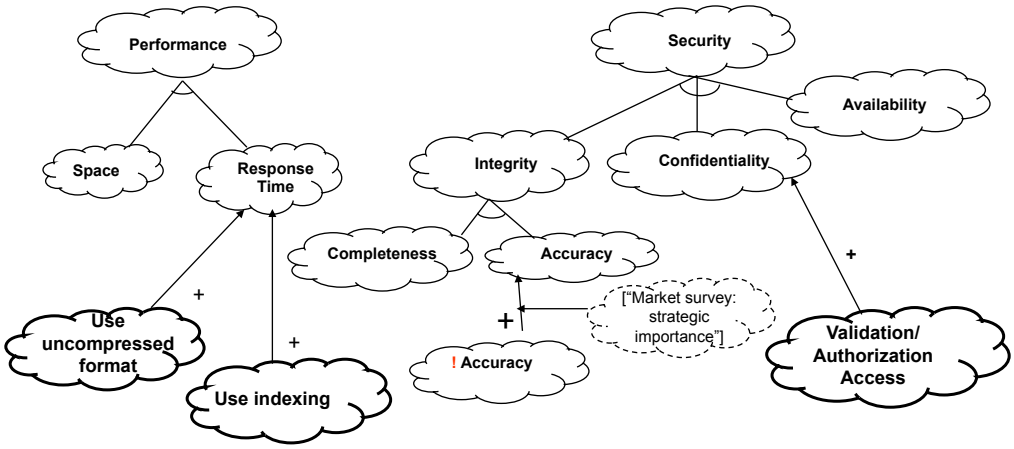


25



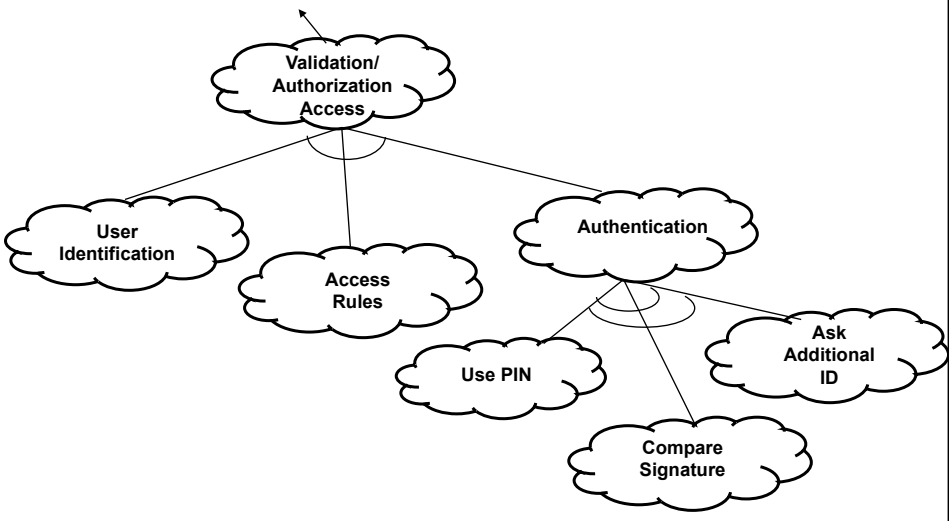
Identify Operationalizations (1)

- Catalogues offer multiple design decisions and implementations for softgoals



26

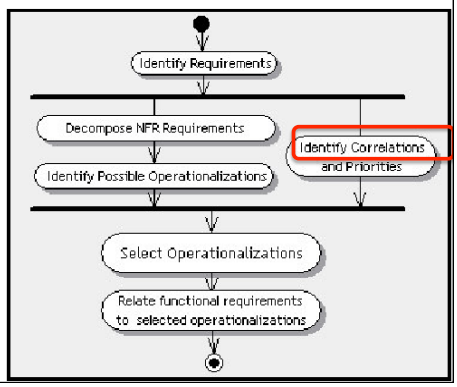
Identify Operationalizations (2)



27

Identify Correlations: contributions, trade-offs

- One of the goals is to identify unexpected impacts of one softgoal on others.
- These contributions should be identified and classified according to their strength:
  - + (positive)
  - ++ (surely positive)
  - - (negative)
  - -- (surely negative)

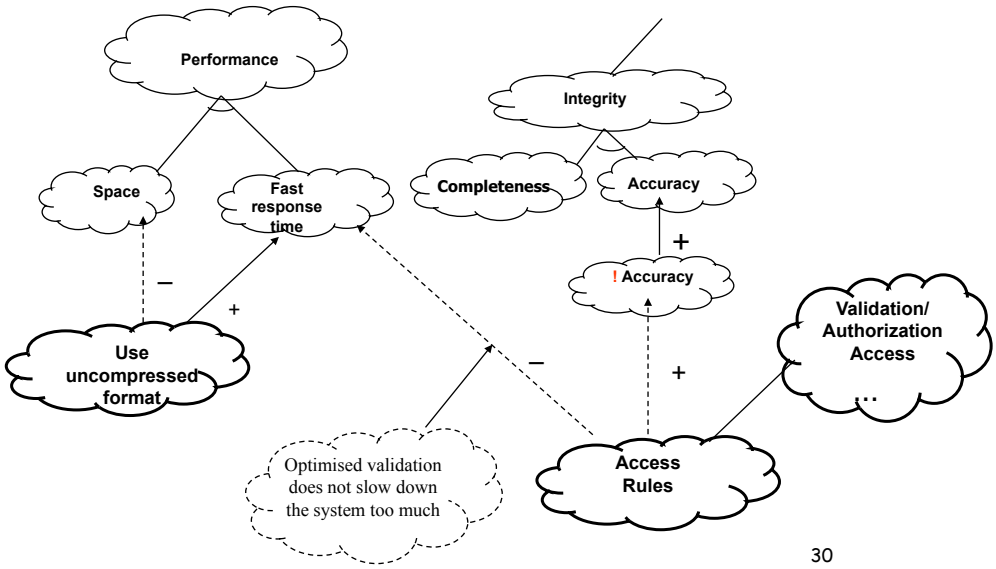


Correlations

- An aim is to identify unexpected impacts on other softgoals:
  - A softgoal may affect another one (not even initially related), highlighting emergent properties
  - The effect may be positive or negative
  - The idenfied correlations (emergent) requires the "merge" of the decomposition graphs

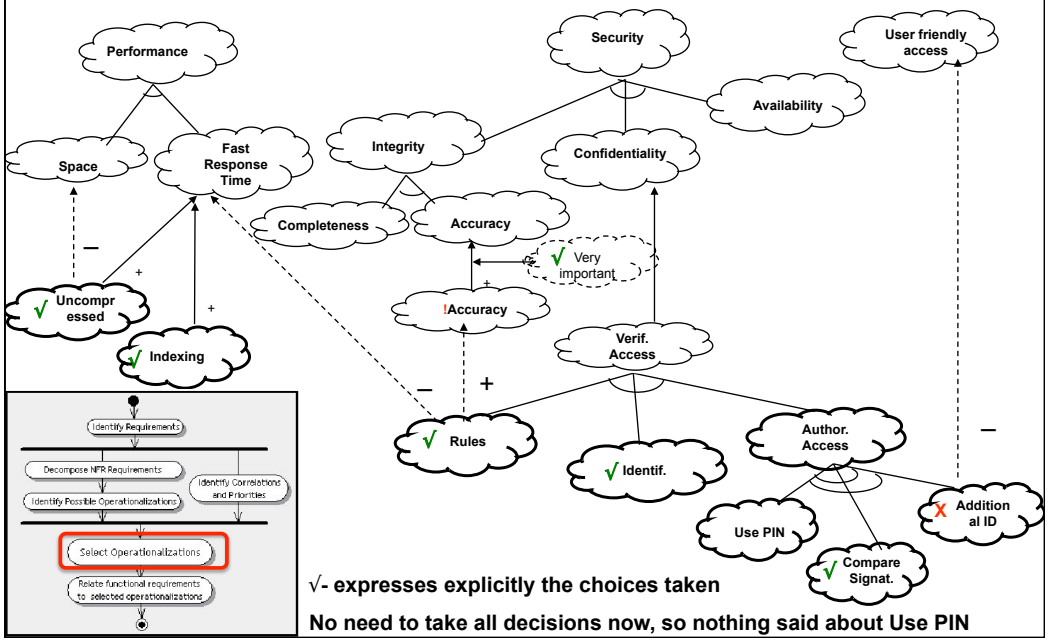
29

Contributions: example



30

Choose between alternatives

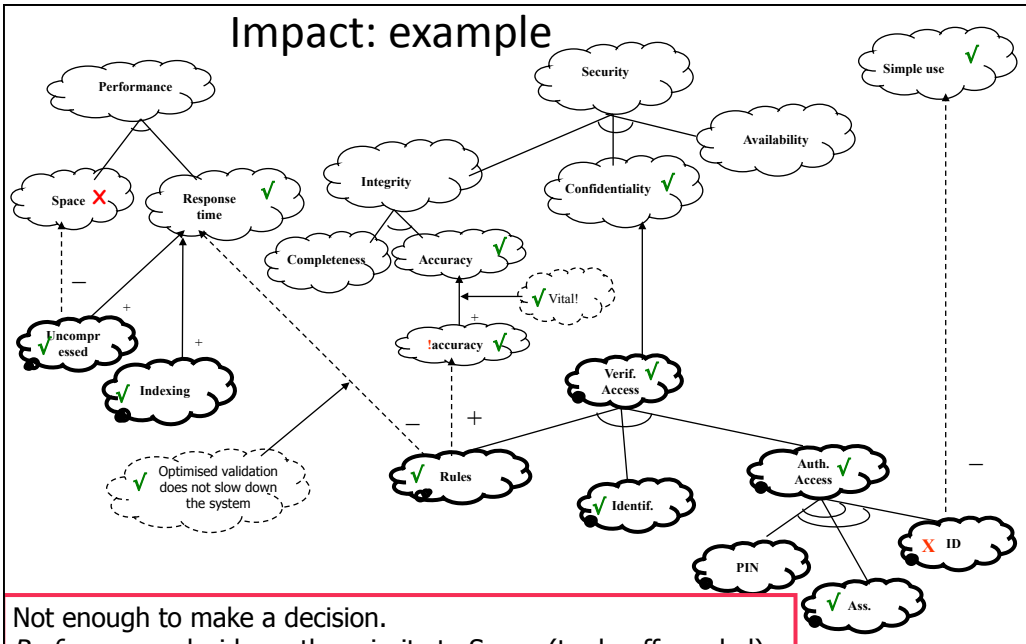


Impact of the choices

- Operationalizations are chosen;
- The objective is to satisfy the main (top) softgoals;
- The consequences of the chosen operationalizations are propagated to the parent softgoals;
- The evaluation of the decisions takes into consideration the contributions of the children softgoals into a parent softgoal;
- Positive and negative contributions are articulated with the choices (accept/reject) to define the path in the tree that better satisfies the stakeholder expectations.

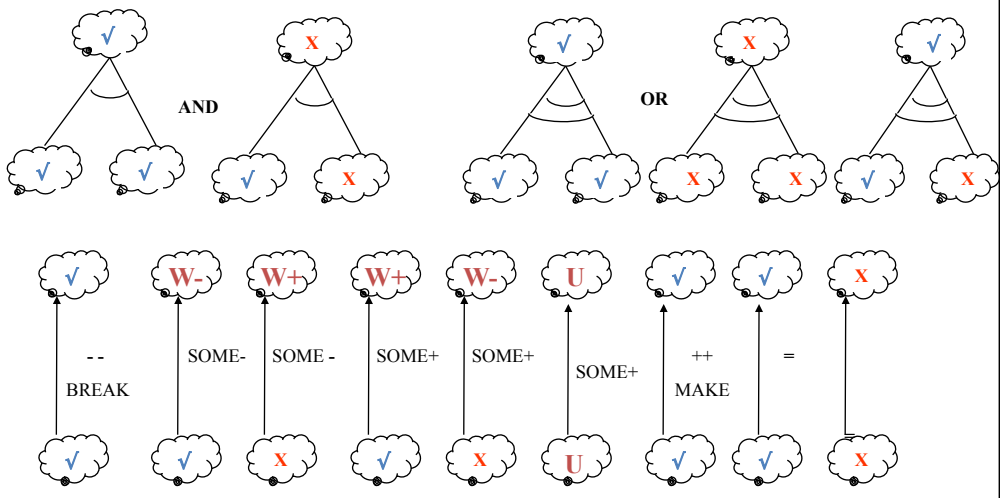
32

Impact: example



Not enough to make a decision.
Performance: decide on the priority to S*pace* (trade-off needed)
Security: needs to be detailed further; only *Confidentiality* and *Simple use* have been satisfied

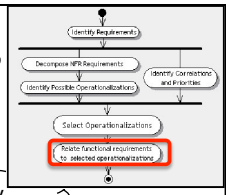
Impact of the choices: rules



"W" = weak; "U" = Undefined

34

Extracting concrete FRs from NFRs



In accounts:

- use indexing + uncompressed format

- Identify users and validate them with rules and authenticate access, comparing them with signatures

35

Positive & negative relationships among NFRs

	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability
Availability							+		+			
Efficiency			−		−	−	−	−		−	−	−
Flexibility		−		−		+	+	+			+	
Integrity		−			−				−		−	−
Interoperability		−	+	−			+					
Maintainability	+	−	+					+			+	
Portability		−	+		+	−			+		+	−
Reliability	+	−	+			+				+	+	+
Reusability		−	+	−	+	+	+	−			+	
Robustness	+	−						+				+
Testability	+	−	+			+		+				+
Usability		−								+	−	

[Wiegers, K. E. (2003). Software Requirements. 2nd edition. Microsoft Press.]

36

Conclusion

Systematic treatment of NFRS during requirements:

- Identifies subjective needs of the project, transforming them into functional requirements
- Highlights hidding shortcommings in a project
- Reuses valuable know-how
- Decreases the need of reengineering a project too early (due to unsatisficed client)
- … and,very importantly, helps identifying early trade-offs

37

Bibliografia

- L.Chung, B.Nixon, E.Yu and J. Mylopoulos "Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers. ISBN 0-7923-8666-3. (2000)
- Boehm, Barry e In, Hoh. *Identifying Quality-Requirement Conflicts*. IEEE Software, March 1996, pp. 25-35
- Chung L., "Representing and Using Non-Functional Requirements: A Process Oriented Approach" Ph.D. Thesis, Dept. of Comp.Sci. University of Toronto, June 1993. Also tech. Rep. DKBS-TR-91-1.

38