Chapter 9, Non-functional Requirements

Organizational
Requirements
Engineering

Prof. Dr. Armin B. Cremers
Sascha Alda

Overview of remaining sessions

- 11.1.2006 Non-functional Requirements
- 18.1.2006 Interactive Systems
- 25.1.2006 Practice Talk “Deutsche Post World Net”
- 1.2.2006 Practice Talk “sd&m”

- 8.2.2006 Written Exam

Overview on this session

- Introduction and Motivation
- Classification and Presentation of Non-functional Requirements
  - Excurse: Tailorability
- Derivation of Non-functional Requirements
- Question catalogue

Non-functional requirements (NFR)

- Non-functional requirements define the overall qualities or attributes of the resulting system
- Non-functional requirements place restrictions on the product being developed, the development process, and specify external constraints that the product must meet.
- Examples of NFR include safety, security, usability, reliability and performance requirements.

- Project management issues (costs, time, schedule) are often considered as non-functional requirements as well

  → not handled in this session

Functional vs. Non-functional requirements

- There is no a clear distinction between functional and non-functional requirements.
- Whether or not a requirement is expressed as a functional or a non-functional requirement may depend:
  - on the level of detail to be included in the requirements document
  - the comprehension of application domain and the desired system
  - experience of developers

Functional vs. Non-functional requirements

- Some properties of a system may be expressed either as a functional or non-functional property.
- Example. The system shall ensure that data is protected from unauthorised access.
- Conventionally a non-functional requirement (security) because it does not specify specific system functionality
- Expressed as functional requirement:
  - 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.

→ Non-functional requirements may result in new functional requirements statements

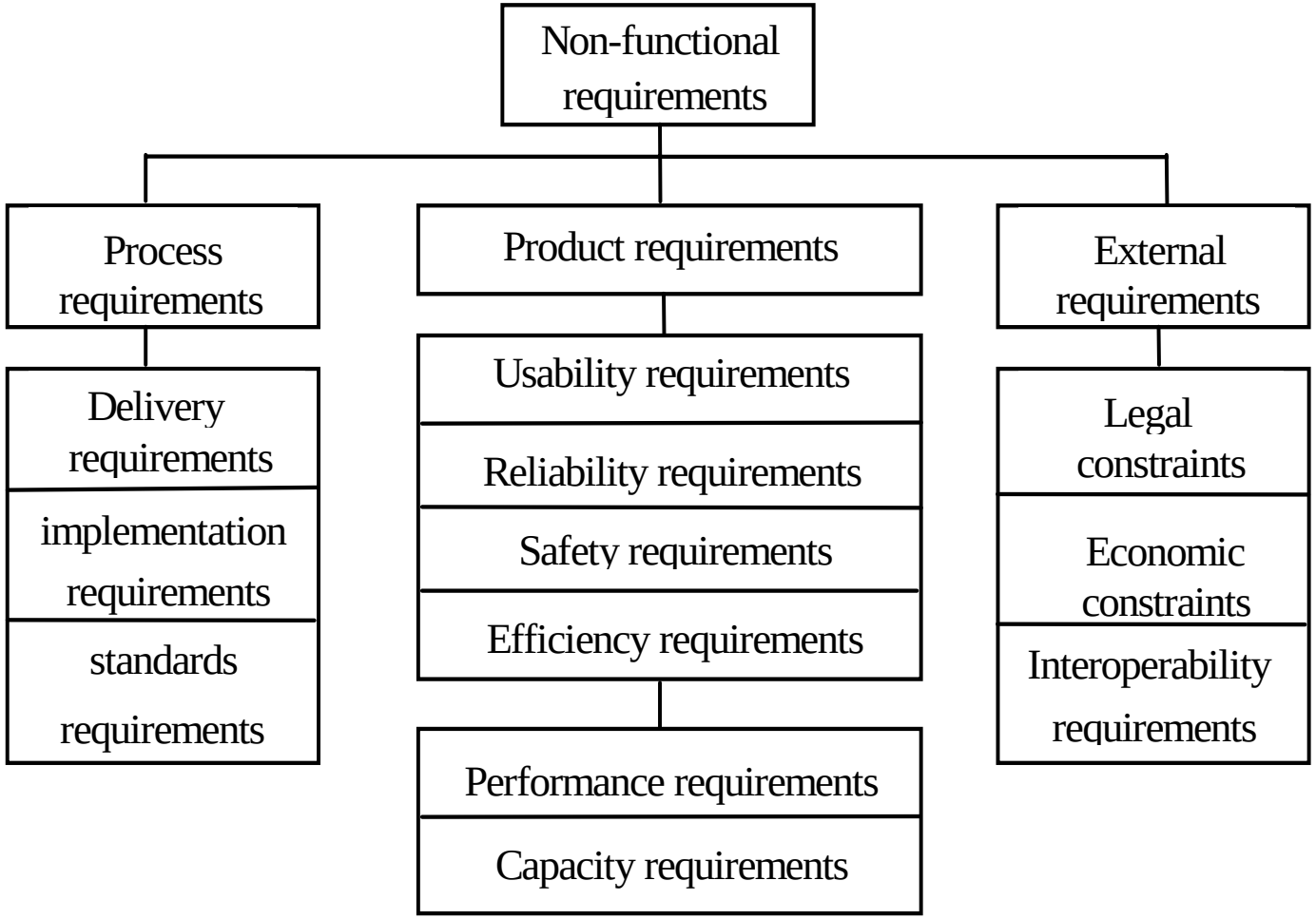
Types of Non-functional requirements

- The ‘IEEE-Std 830 - 1993’ lists 13 non-functional requirements to be included in a Software Requirements Document.
  - Performance requirements
  - Interface requirements
  - Operational requirements
  - Resource requirements
  - Verification requirements
  - Acceptance requirements
  - Documentation requirements
  - Security requirements
  - Portability requirements
  - Quality requirements
  - Reliability requirements
  - Maintainability requirements
  - Safety requirements

Classification of Non-functional requirements (Jacobson, 1999)

- The FURPS+ model is proposed for the Unified Process.
  - Functional Requirements
  - Usability
  - Reliability
  - Performance
  - Supportability
- The FURPS+ model provides additional requirements typically also included under the general label of non-functional requirements:
  - Implementation requirements
  - Interface requirements
  - Operations requirements
  - Packaging requirements
  - Legal requirements

Classification of Non-functional requirements (Sommerville)



Product Requirements

- Specify the desired characteristics that a system or subsystem must possess.
- Most NFRs are concerned with specifying constraints on the behaviour of the executing system.
  - Limit the freedom of the system designers
  - Limit the free choice of off-the-shelf-components

- Some can be formulated precisely and easily quantified
  - Performance, Reliability
- Some are difficult to quantify
  - Usability, Adaptability

Requirements for critical systems

- Non-functional (product) requirements play an important role for critical systems
- Systems whose ‘failure’ causes significant economic, physical or human damage to organisations or people.
- There are three principal types of critical system:
  - Business critical systems
    - ➔ Failure leads to significant economic damage
  - Mission critical systems
    - ➔ Failure leads to the abortion of a mission
  - Safety critical systems
    - ➔ Failure endangers human life

Requirements for critical systems

- The principal non-functional constraints which are relevant to critical systems:
  - Reliability
  - Performance
  - Security
  - Safety
  - Usability

Reliability

- Reliability is the ability of a system to perform its required functions under stated conditions for a specific period of time
- Constraints on the run-time behaviour of the system
- Can be considered under two separate headings:
  - Availability - is the system available for service when requested by end-users.
  - Failure rate - how often does the system fail to deliver the service as expected by end-users.
- Many related items:
  - Dependability
    - Dependability of a computing system is the ability to deliver service that can justifiably be trusted by users
  - Reputation
    - the general opinion (more technically, a social evaluation) of the public toward a person, a group of people

Performance

- Performance requirements concern the *speed of operation* of a system
- Types of performance requirements:
  - Response requirements (how quickly the system reacts to a user input)
  - Throughput requirements (how much the system can accomplish within a specified amount of time)
  - Availability requirements (is the system available for service when requested by end-users)
- Many related items:
  - Scalability
    - the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added

Product Requirements
Examples

- The System service X shall have an availability of 999/1000 or 99%. This is a *reliability* requirement which means that out of every 1000 requests for this service, 999 must be satisfied.

- System Y shall process a minimum of 8 transactions per second. This is a *performance* requirement.

Security

- Security requirements are included in a system to ensure:
  - Unauthorised access to the system and its data is not allowed
  - Ensure the integrity of the system from accidental or malicious damage
- Examples of security requirements are:
  - The access permissions for system data may only be changed by the system's data administrator
  - All system data must be backed up every 24 hours and the backup copies stored in a secure location which is not in the same building as the system
  - All external communications between the system's data server and clients must be encrypted

Usability

- Usability is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of system or component
- Usability requirements include:
  - Well-structured user manuals
  - Informative error messages
  - Help facilities
  - Well-formed graphical user interfaces

Usability Metrics

- Measurable attributes of usability requirements include:
  - *Entry requirements* Measured in terms of years of experience with class of applications or simply age
  - *Learning requirements* Denotes the time needed to learn the facilities of the system. This could be measured in terms of speed of learning, say hours of training required before independent use is possible
  - *Handling requirements* Denotes the error rate of the end-users of the system. This could be measured in terms of the errors made when working at normal speed
  - *Likeability* Denotes 'niceness' to use. The most direct to measure user satisfaction is to survey actual users and record the proportion who 'like to work with the product'

- ... more about Usability see next week

Safety

- No consensus in the system's engineering community about what is meant by the term 'safety requirement'

- Usage of the term often depends on the culture and practice of the organisation (often mixed up with security)
- Informal definition:

  Safety requirements are the 'shall not' requirements which exclude unsafe situations from the possible solution space of the system

Examples of safety requirements

- The violated system shall not permit any further operation unless the operator guard is in place
- The system shall not operate if the external temperature is below 4 degrees Celsius
- The system should not longer operate in case of fire (e.g. an elevator)
- The system should no longer operate if security attacks have become obvious (→ relation to security requirements)

Supportability

- **Supportability** requirements are concerned with the ease of changes to the system after deployment. Classes:
- Adaptability:
  - The ability to change the system to deal with additional application domain concepts (adaptivity = autonomous adaptation)
- Maintainability:
  - The ability to change the system to deal with new technology or to fix defects)
- Internationalization:
  - The ability to change the system to deal with additional international conventions such as languages, or number formats, styles)
- Portability:
  - The ease with which a system or component can be transferred from one environment to another)
- Tailorability:
  - End-User Adaptation (next slides)

Excurse: Tailorability

- Fields of application are differentiated and dynamically changing
  - New tasks and working contexts arise
  - Individual qualifications
- Tailorability is defined
  - changing aspects of an application‘s functionality
  - in a persistent way (by means of tailored artifacts)
  - during the use of an application (at runtime)
  - by end-users or local experts
- Technical flexibility beyond
  - modifications of parameters (e.g. Windows Registry)
  - (re-)programming

Tailorability
Challenges

- Flexible Architecture
  - Rule-based architectures
  - Component-based architectures (simple and compound components)

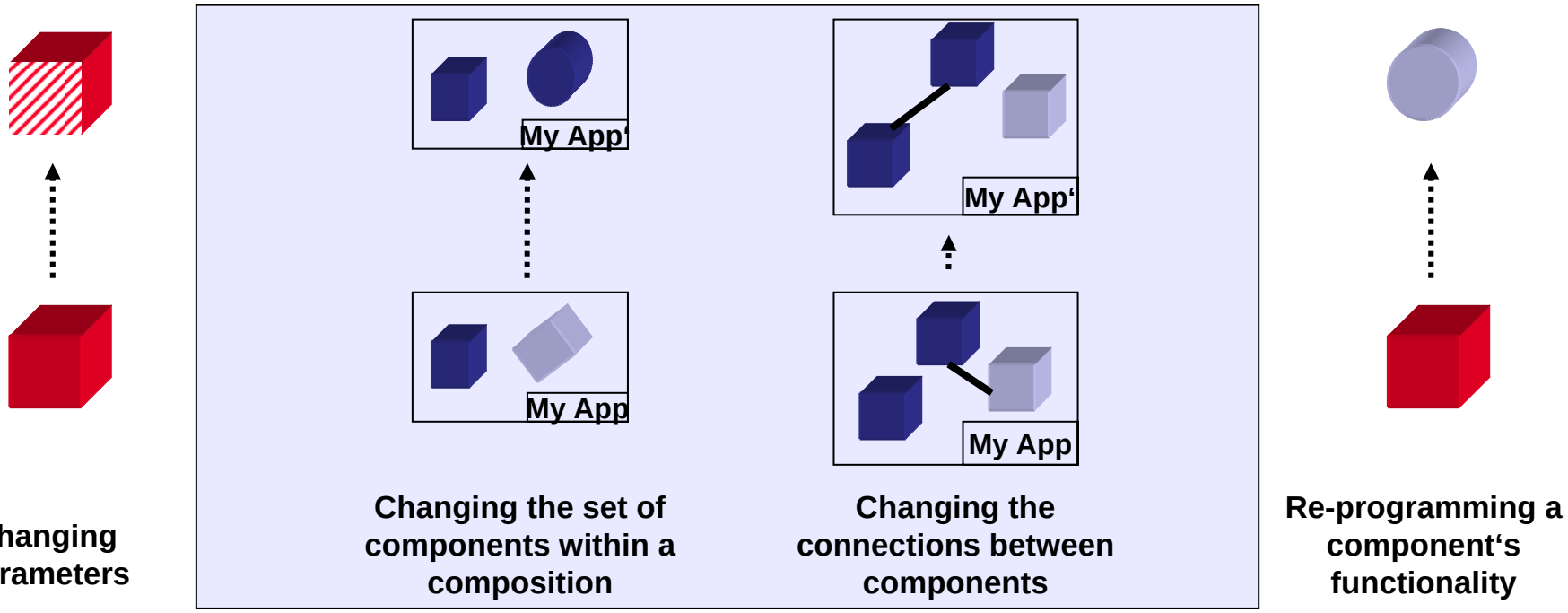
- Appropriate Interfaces
  - Visualizing and manipulating tailored artifacts: 2D and 3D Interfaces
  - Describing tailored artifacts: Annotations, attaching examples
  - Understanding tailored artifacts: Exploration Environments
  - Providing support for tailoring: Integrity checking
  - Accessing tailoring functions: Direct Activation

Tailorability
Levels of tailoring strategies (Morch, 1997)

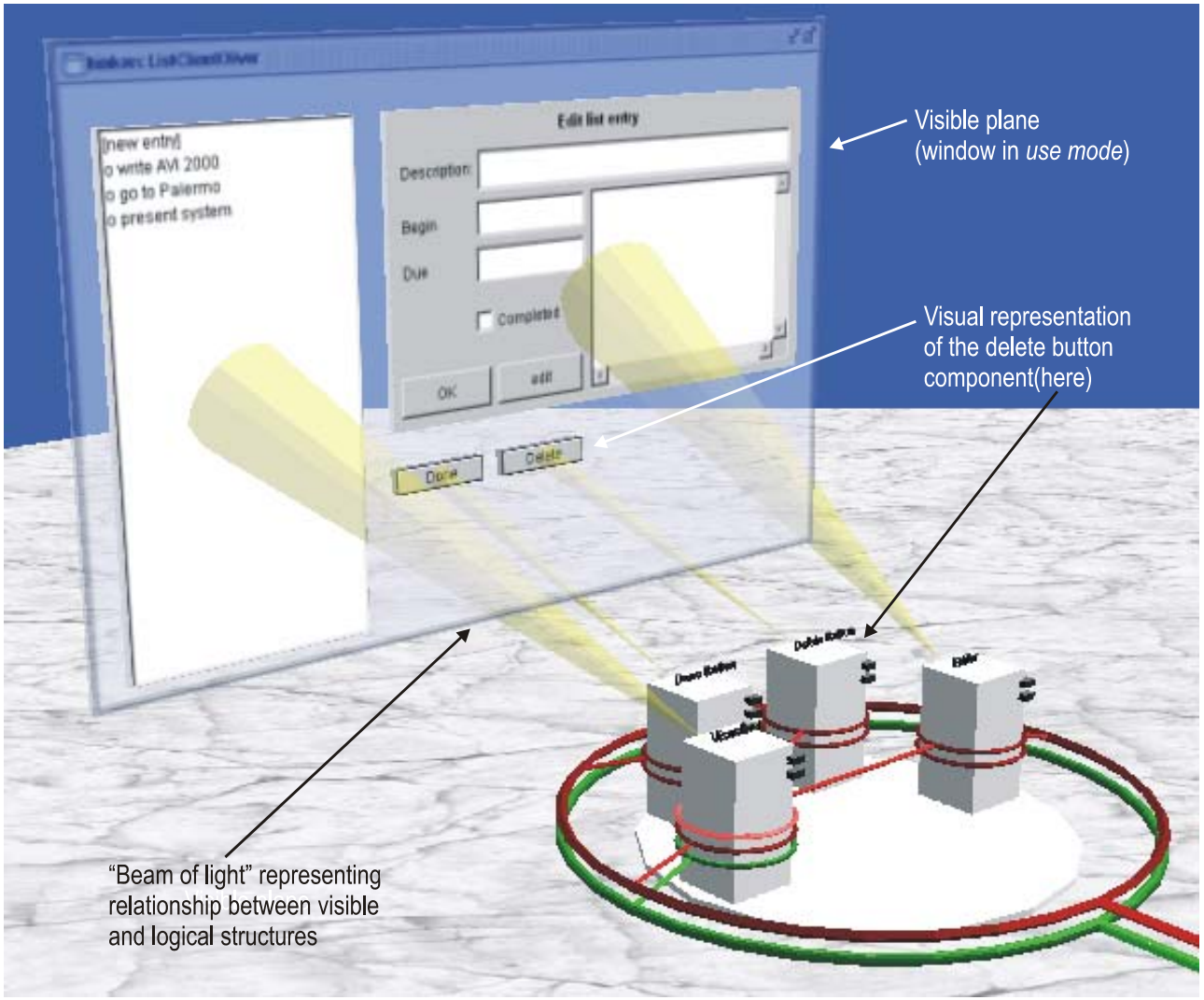
- Problem: User do not have experience with tailoring
- Goal: Provide tailoring routines on different levels of complexity
- End-user with little experience can adopt to less complex routines
- More experienced may use more sophisticated routines
- Proposed Levels:
  - Customization
    - ➔ Modification of presentation objects among a set of pre-defined configuration options
  - Integration
    - ➔ Creation or the combination of (existing) program behavior that results in new behavior
  - Extension
    - ➔ Adding completely new behavior (→ re-implementation)

Component-Orientation for as basis for tailorable systems

- Properties of components
  - ➔ Independently developed parts of software
  - ➔ Independently exchangeable
  - ➔ Several components interact as one application or system



Example for 3D tailoring environment
(For client-server-based groupware system)



Product Requirements
Examples

- There are product requirements which relate to the source code of the system
- Examples
  - The system shall be developed for PC and Macintosh platforms. This is a portability requirement which affects the way in which the system may be designed
  - The system must encrypt all external communications using the RSA algorithm. This is a security requirement which specifies that a specific algorithm must be used in the product

Product Requirements
Conflicts

- Product requirements are often conflict. For example:
  - A requirement for a certain level of performance may be contradicted by security requirements which use processor capacity to carry out complex cipher algorithms
  - A requirement on the memory utilisation of the system may be contradicted by another requirement which specifies that a standard compiler which does not generate compact code must be used
  - Adaptable or tailorable software may lead to software that is unsafe
- The process of arriving at a trade-off in these conflicts depends on:
  - The level importance attached to the requirement
  - The consequence of the change on the other requirements and,
  - The wider business goals

Process Requirements

- Process requirements are constraints placed upon the development process of the system
- Process requirements include:
  - Requirements on development standards and methods which must be followed
  - CASE tools which should be used
  - The management reports which must be provided

Examples of process requirements

- The development process to be used must be explicitly defined and must be conformant with ISO 9000 standards
- The system must be developed using the XYZ suite of CASE tools
- Management reports setting out the effort expended on each identified system component must be produced every two weeks
- A disaster recovery plan for the system development must be specified

External requirements

- May be placed on both the product and the process
- Derived from the environment in which the system is developed
- External requirements are based on:
  - application domain information
  - organisational considerations
  - the need for the system to work with other systems
  - health and safety or data protection regulations

- Legal requirements:
  - Concerned with licensing, regulation, and certification issue

External requirements

- Medical data system: The organization's data protection officer must certify that all data is maintained according to data protection legislation before the system is put into operation.

- A software is regulated under the GNU General Public License
  - make sure the software is free for all its users, that is, everybody can share and change the software
  - No warranty, no patents

- Web pages developed for federal government of NRW must comply with the law for equality of treatment of people with disabilities (obstacle free Internet, from 1/1/2006)
  - Blind people should be enabled to interpret web pages (tools...)

Deriving non-functional requirements

- No adequate methods are provided
  - Functional analysis or object-oriented analysis are limited as soon as non-functional concerns are regarded
- NFRs are difficult to express. Reasons:
  - Certain constraints are related to the design solution that is unknown at the requirements stage
  - Certain constraints, are highly subjective and can only be determined through complex empirical evaluations
  - Non-functional requirements tend to be related to one or more functional requirements
  - Non-functional requirements tend to conflict and contradict
  - There are no ‘universal’ rules and guidelines for determining when non-functional requirements are optimally met.

Stakeholder concerns

- Stakeholders normally have a number of ‘concerns’ that come with their desired system
- Concerns are typically high-level strategic goals and are non-functional
- Examples include:
  - Critical business objectives (standards)
  - Essential system characteristics (e.g. security)
  - Safety, performance, functionality and maintainability
- Vaguely defined user concerns may be related to NFRs

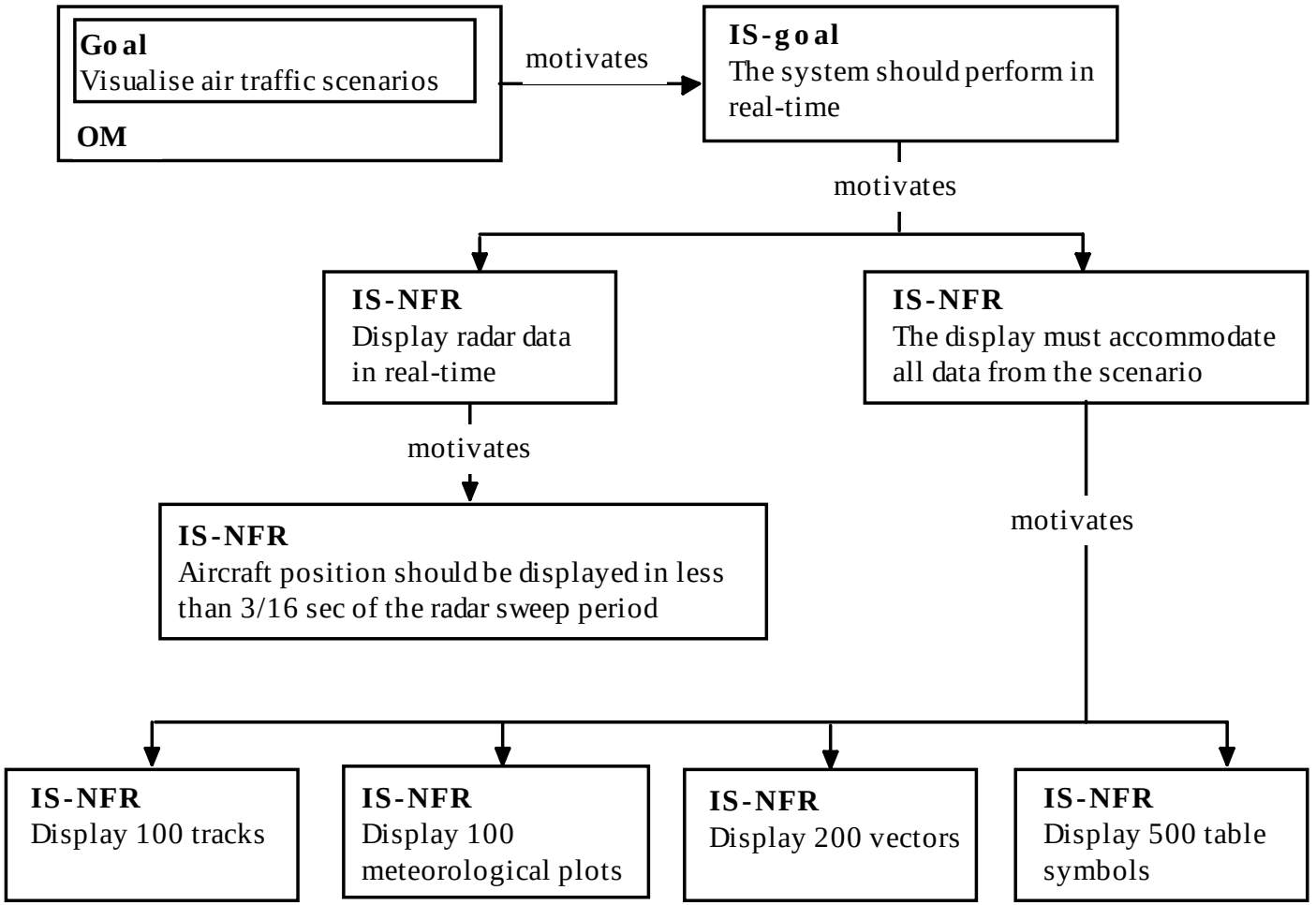
Relationships between user needs, concerns and NFRs

User's need	User's concern	Non-functional requirement
Function	1. Ease of use 2. Unauthorised access 3. Likelihood of failure	1. Usability 2. Security 3. Reliability
Performance	1. Resource utilisation 2. Performance verification 3. Ease of interfacing	1. Efficiency 2. Verifiability 3. Interoperability
Change	1. Ease of repair 2. Ease of change 3. Ease of transport ? 4. Ease of expanding or upgrading capacity or performance ?	1. Maintainability 2. Flexibility 3. Portability 4. Expandability

Goal-based derivation

- Relates non-functional requirements to the goals of the enterprise
- Goal-based NFR derivation is a 3 step approach:
  - Identify the enterprise goal
  - Decompose of the goal into sub-goals
  - Identify non-functional requirements.

Example of goal-based derivation



Testable NFRs

- Stakeholders may have vague goals which cannot be expressed precisely
- Vague and imprecise ‘requirements’ are problematic
- NFRs should satisfy two attributes
  - Must be objective
  - Must be testable (Use measurable metrics)
- It is not always possible to express NFRs objectively

Examples of measurable metrics for Non-functional requirements

Property	Metric
Performance	1. Processed transactions per second 2. Response time to user input
Reliability	1. Rate of occurrence of failure 2. Mean time to failure
Availability	Probability of failure on demand
Size	Kbytes, Mbytes
Usability	1. Time taken to learn the software 2. Number of errors made by user
Robustness	Time to restart the system after failure
Portability	Number of target systems

Nonfunctional Requirements:
Trigger Questions (1/4)

- User interface and human factors
  - What type of user will be using the system?
  - Will more than one type of user be using the system?
  - What sort of training will be required for each type of user?
  - Is it particularly important that the system be easy to learn?
  - Is it particularly important that users be protected from making errors?
  - What sort of input/output devices for the human interface are available, and what are their characteristics?
- Documentation
  - What kind of documentation is required?
  - What audience is to be addressed by each document?
- Hardware considerations
  - What hardware is the proposed system to be used on?
  - What are the characteristics of the target hardware, including memory size and auxiliary storage space?

Nonfunctional Requirements:
Trigger Questions (2/4)

- Performance characteristics
  - Are there any speed, throughput, or response time constraints on the system?
  - Are there size or capacity constraints on the data to be processed by the system?
- Error handling and extreme conditions
  - How should the system respond to input errors?
  - How should the system respond to extreme conditions?
- System interfacing
  - Is input coming from systems outside the proposed system?
  - Is output going to systems outside the proposed system?
  - Are there restrictions on the format or medium that must be used for input or output?

Nonfunctional Requirements: Trigger Questions (3/4)

- Quality issues
  - What are the requirements for reliability?
  - Must the system trap faults?
  - What is the maximum time for restarting the system after a failure?
  - What is the acceptable system downtime per 24-hour period?
  - Is it important that the system be portable (able to move to different hardware or operating system environments)?
- System Modifications
  - What parts of the system are likely candidates for later modification?
  - What sorts of modifications are expected (levels of adaptation)?
  - Are the users willing to tailor an application?
  - What kind of interface is required?
  - Might unwary adaptations lead to unsafe system states?

Nonfunctional Requirements:
Trigger Questions (4/4)

- Security Issues
  - Must access to any data or the system itself be controlled?
  - Is physical security an issue?

- Resources and Management Issues
  - How often will the system be backed up?
  - Who will be responsible for the back up?
  - Who is responsible for system installation?
  - Who will be responsible for system maintenance?

Summary

- Non-functional requirements define the overall qualities or attributes of the resulting system
- Non-functional requirements may be classified into three main types; product, process and external requirements
- Product requirements specify the desired characteristics that the system or subsystem must posses
- Non-functional requirements tend to conflict and interact with other system requirements
- The principal non-functional constraints which are relevant to critical systems are reliability, performance, security, usability and safety