Requirements

- Functional: describe the interactions between the system and its environment, independently from the implementation
- Non-functional: measurable/perceivable properties of the systems not directly related to functional aspects

Requirements specification

- Textual description of system behaviour
- Basic specification technique
- Most used in practice
- ISO/IEC/IEEE standard 29148:2011 (E)
- Should be accessible from the IEEE digital library (*https://ieeexplore.ieee.org/*)
- (slides partly based on 5.2 “Requirements fundamentals”)
- … I shall encourage you to read it !

Goal of a set of requirements

- enables an agreed understanding between stakeholders
  acquirers, users, customers, operators, suppliers
- is validated against real-world needs, can be implemented
- provides a basis of verifying designs and accepting solutions
- start with stakeholders intentions
- needs, goals, or objectives
- iterative process from stakeholders to system requirements

Well-formed requirements

- can be verified,
- has to be met or possessed by a system to solve a stakeholder problem or to achieve a stakeholder objective,
- is qualified by measurable conditions and bounded by constraints,
- defines the performance of the system when used by a specific stakeholder or the corresponding capability of the system, but not a capability of the user, operator, or other stakeholder.

What is a requirement, actually

- is a statement expressing a need and its associated constraints and conditions
- is written in natural language
  - **structural language, “semi-formal”**
- it comprises a subject, a verb, a complement
  - **subject of the requirement**
  - **what shall be done**

Some syntax example (1)

- **[Condition][Subject][Action][Object][Constraint]**

- When signal x is received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint].

Some syntax example (2)

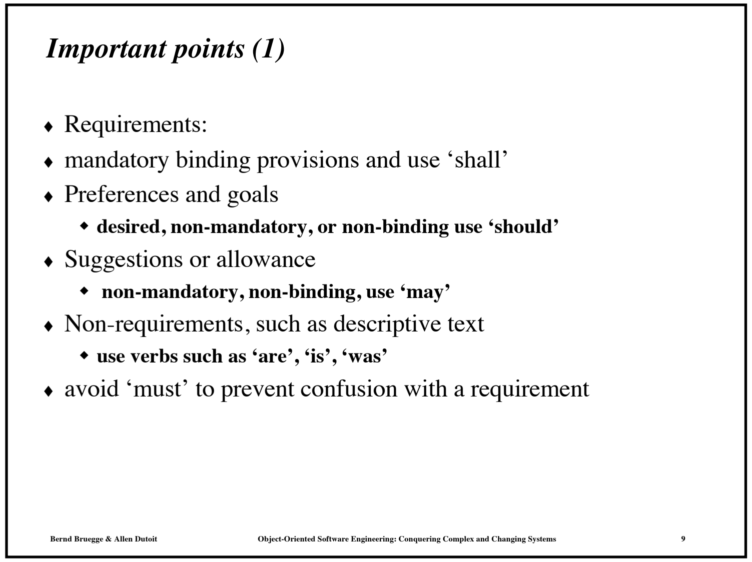
- **[Condition][Subject][Action][Object][Constraint]**

- At sea state 1 [Condition], the Radar System [Subject] shall detect [Action] targets [Object] at ranges out to 100 nautical miles [Constraint].

Some syntax example (3)

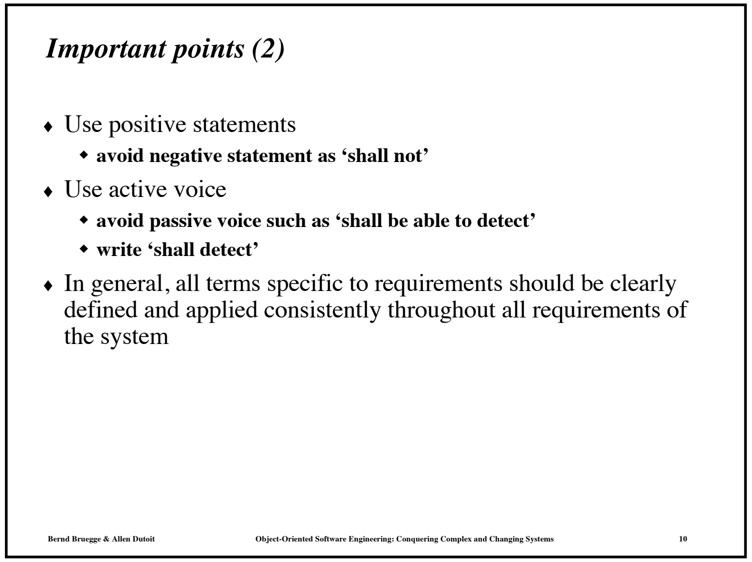
- **[Subject][Action][Object][Constraint]**

- The invoice system [Subject], shall display [Action] pending customer invoices [Object] in ascending order in which invoices are to be paid [Constraint].



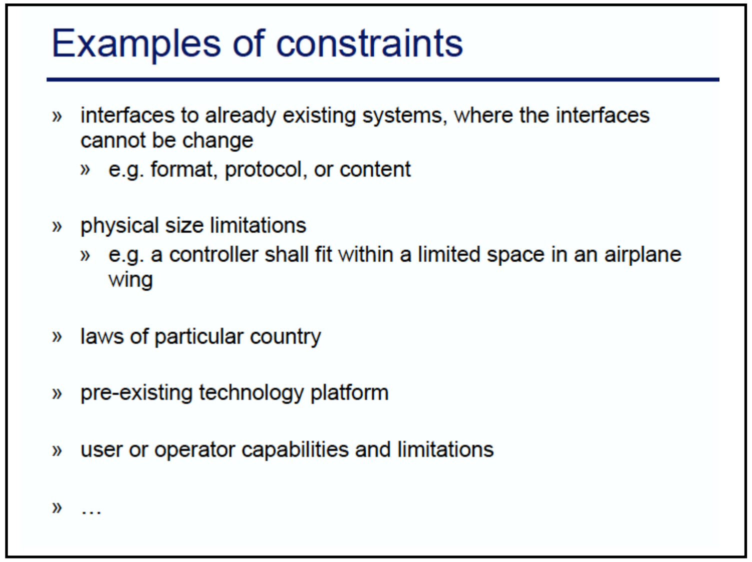
Important points (1)

- Requirements:
- mandatory binding provisions and use ‘shall’
- Preferences and goals
  - **desired, non-mandatory, or non-binding use ‘should’**
- Suggestions or allowance
  - **non-mandatory, non-binding, use ‘may’**
- Non-requirements, such as descriptive text
  - **use verbs such as ‘are’, ‘is’, ‘was’**
- avoid ‘must’ to prevent confusion with a requirement



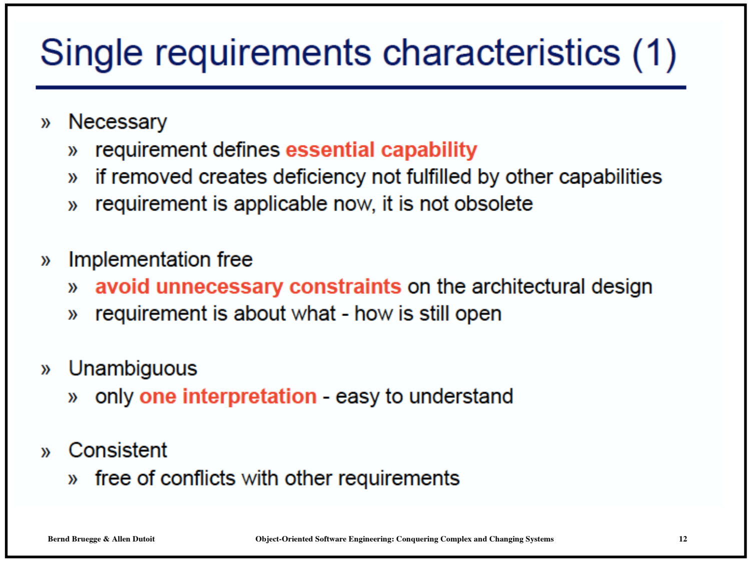
Important points (2)

- Use positive statements
  - **avoid negative statement as ‘shall not’**
- Use active voice
  - **avoid passive voice such as ‘shall be able to detect’**
  - **write ‘shall detect’**
- In general, all terms specific to requirements should be clearly defined and applied consistently throughout all requirements of the system



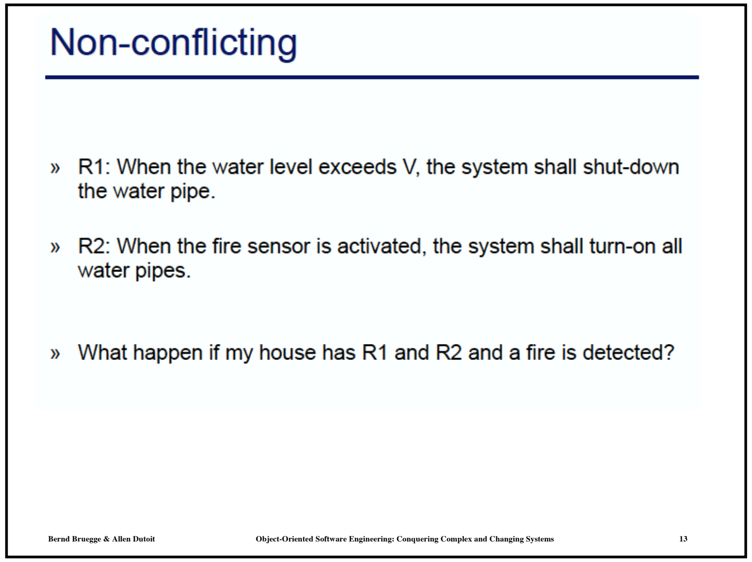
Examples of constraints

- » interfaces to already existing systems, where the interfaces cannot be change
  - » e.g. format, protocol, or content
- » physical size limitations
  - » e.g. a controller shall fit within a limited space in an airplane wing
- » laws of particular country
- » pre-existing technology platform
- » user or operator capabilities and limitations
- » …



Single requirements characteristics (1)

- » Necessary
  - » requirement defines **essential capability**
  - » if removed creates deficiency not fulfilled by other capabilities
  - » requirement is applicable now, it is not obsolete
- » Implementation free
  - » **avoid unnecessary constraints** on the architectural design
  - » requirement is about what - how is still open
- » Unambiguous
  - » only **one interpretation** - easy to understand
- » Consistent
  - » free of conflicts with other requirements

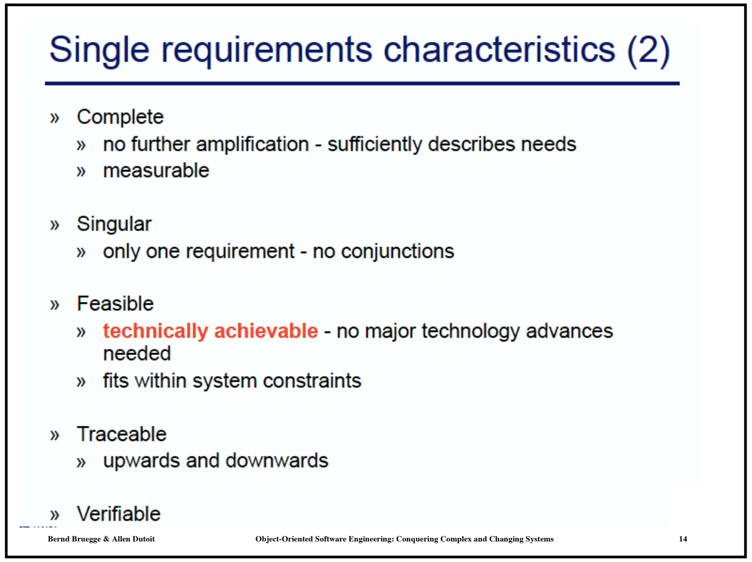


Non-conflicting

» R1: When the water level exceeds V, the system shall shut-down the water pipe.

» R2: When the fire sensor is activated, the system shall turn-on all water pipes.

» What happen if my house has R1 and R2 and a fire is detected?



Single requirements characteristics (2)

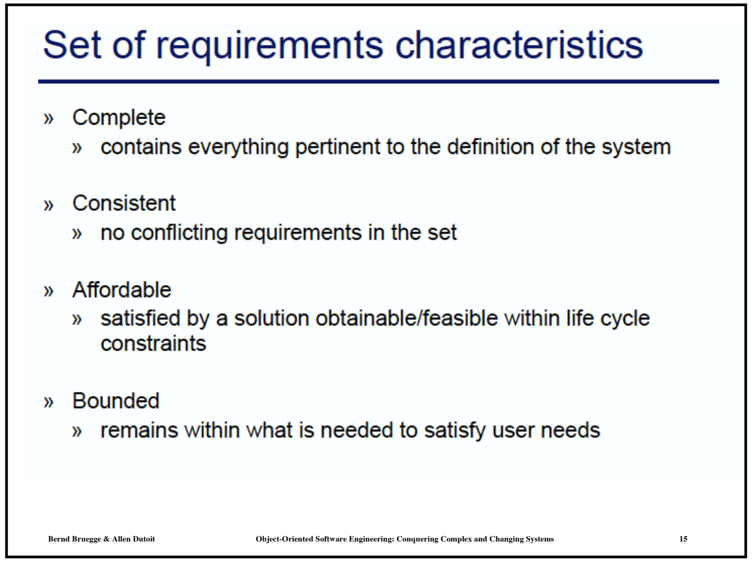
» Complete
  » no further amplification - sufficiently describes needs
  » measurable

» Singular
  » only one requirement - no conjunctions

» Feasible
  » **technically achievable** - no major technology advances needed
  » fits within system constraints

» Traceable
  » upwards and downwards

» Verifiable



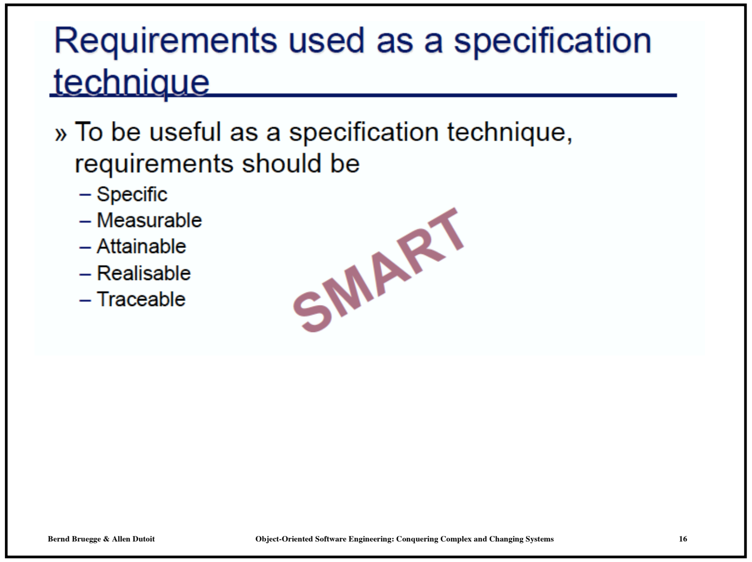
Set of requirements characteristics

» Complete
  » contains everything pertinent to the definition of the system

» Consistent
  » no conflicting requirements in the set

» Affordable
  » satisfied by a solution obtainable/feasible within life cycle constraints

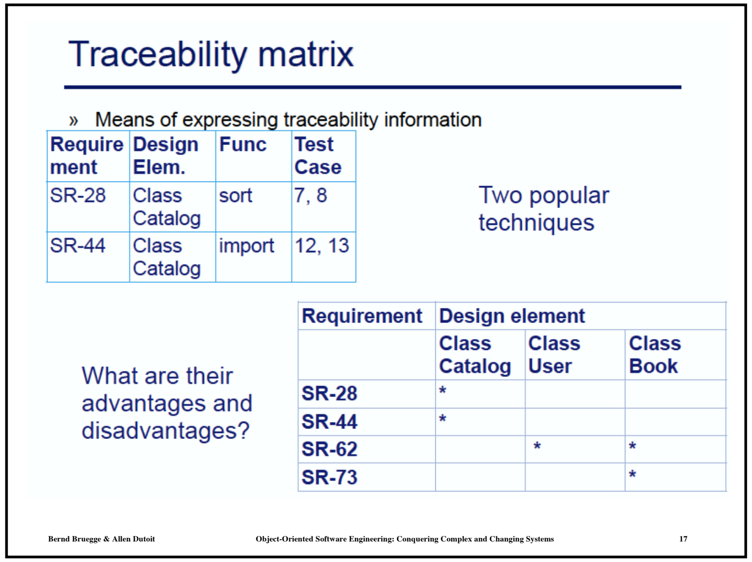
» Bounded
  » remains within what is needed to satisfy user needs



Requirements used as a specification technique

» To be useful as a specification technique, requirements should be
  – Specific
  – Measurable
  – Attainable
  – Realisable
  – Traceable

SMART



Traceability matrix

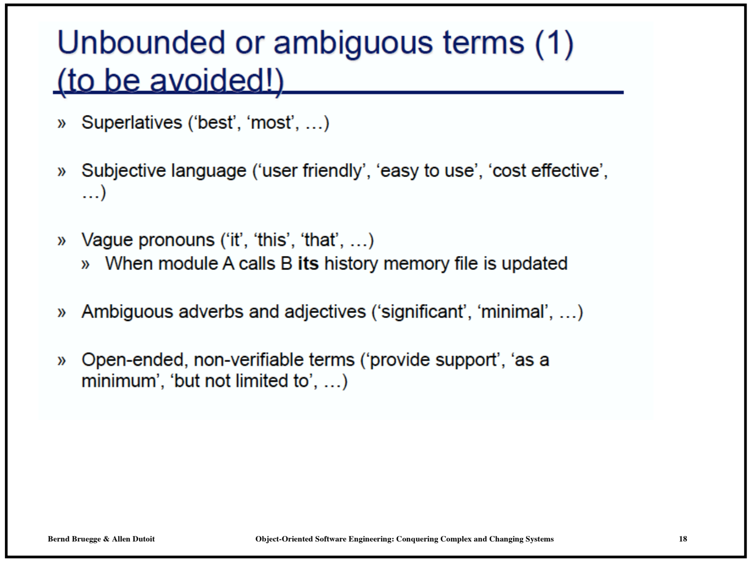
» Means of expressing traceability information

Require ment	Design Elem.	Func	Test Case
SR-28	Class Catalog	sort	7, 8
SR-44	Class Catalog	import	12, 13

Two popular techniques

What are their advantages and disadvantages?

Requirement	Design element		
	Class Catalog	Class User	Class Book
SR-28	*		
SR-44	*		
SR-62		*	*
SR-73			*



Unbounded or ambiguous terms (1) (to be avoided!)

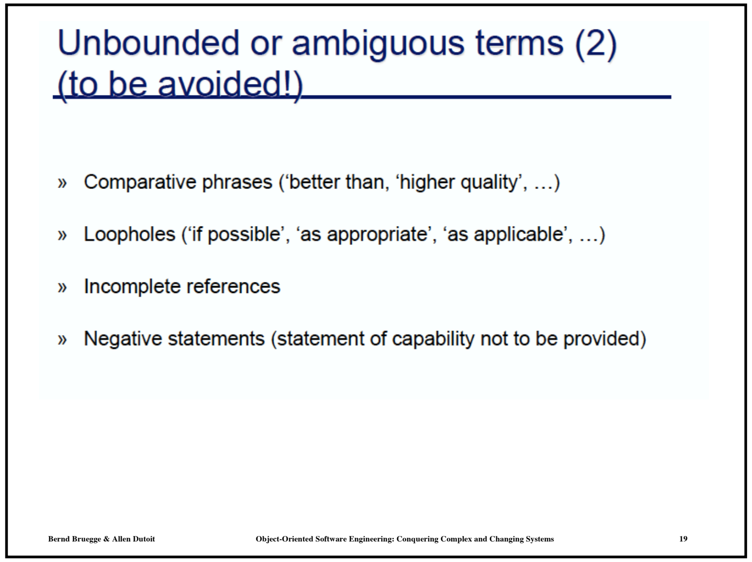
» Superlatives ('best', 'most', …)

» Subjective language ('user friendly', 'easy to use', 'cost effective', …)

» Vague pronouns ('it', 'this', 'that', …)
  » When module A calls B **its** history memory file is updated

» Ambiguous adverbs and adjectives ('significant', 'minimal', …)

» Open-ended, non-verifiable terms ('provide support', 'as a minimum', 'but not limited to', …)



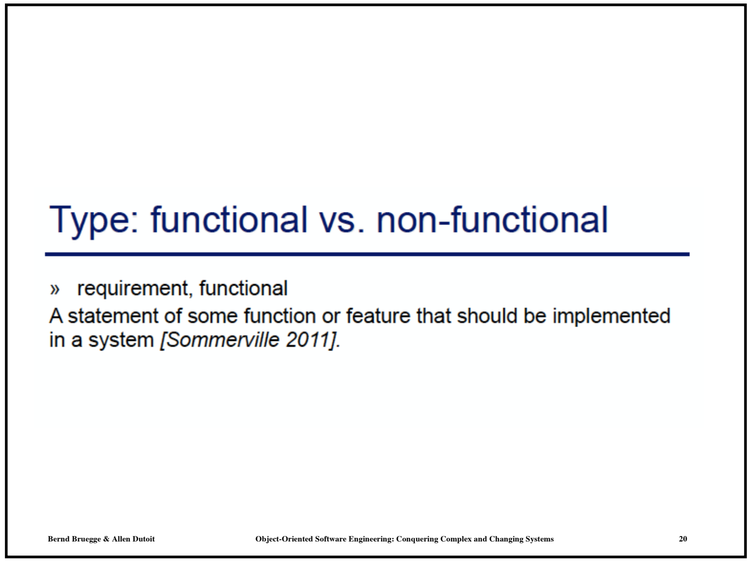
Unbounded or ambiguous terms (2) (to be avoided!)

» Comparative phrases ('better than, 'higher quality', …)

» Loopholes ('if possible', 'as appropriate', 'as applicable', …)

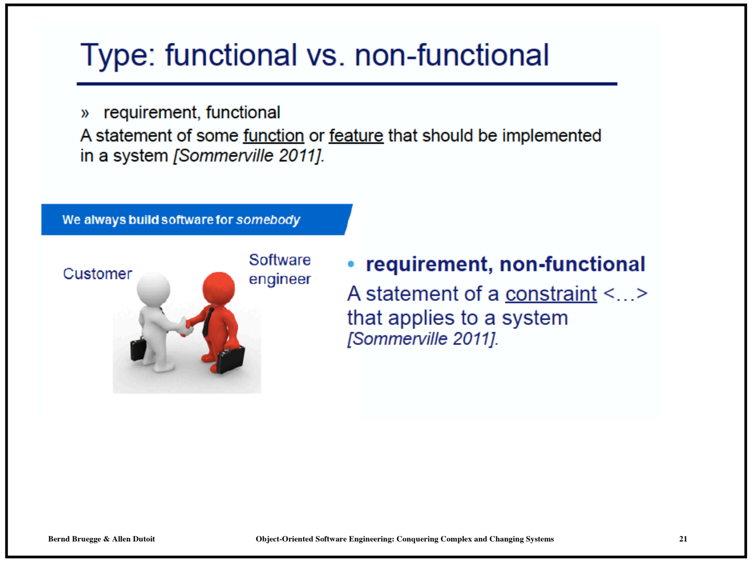
» Incomplete references

» Negative statements (statement of capability not to be provided)



Type: functional vs. non-functional

» requirement, functional
A statement of some function or feature that should be implemented in a system *[Sommerville 2011]*.

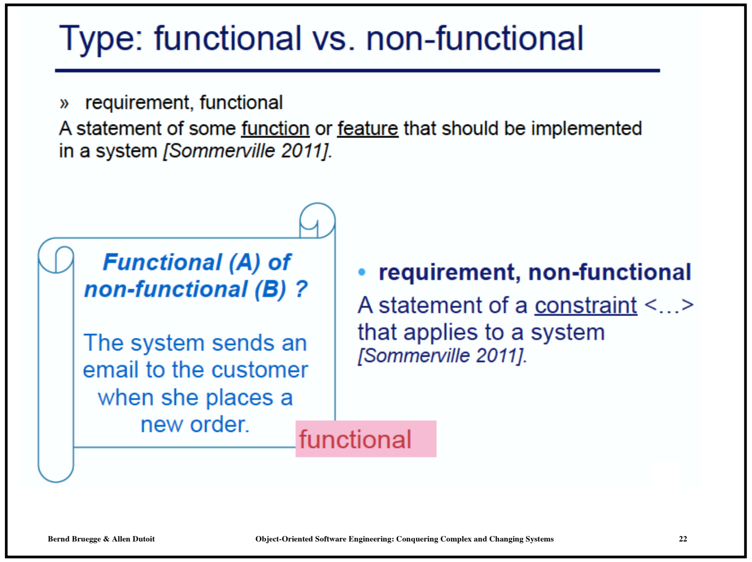


Type: functional vs. non-functional

» requirement, functional
A statement of some <u>function</u> or <u>feature</u> that should be implemented in a system *[Sommerville 2011]*.

We always build software for *somebody*

- **requirement, non-functional**
A statement of a <u>constraint</u> <…> that applies to a system *[Sommerville 2011]*.



Type: functional vs. non-functional

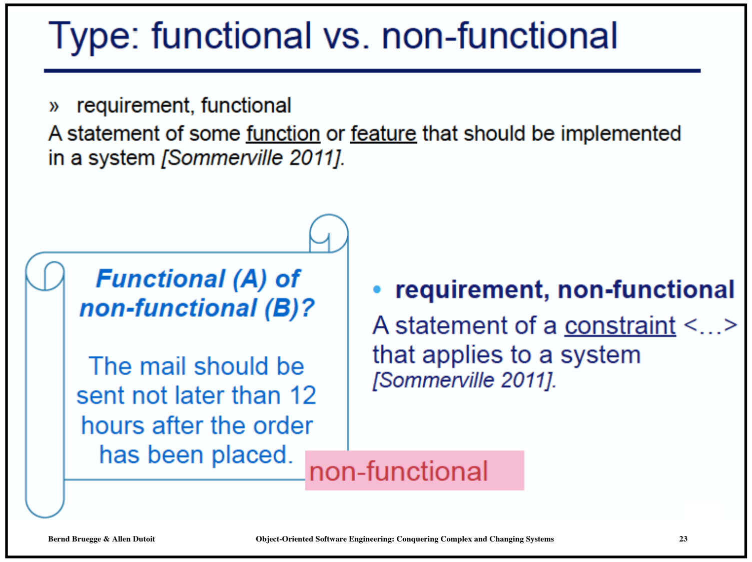
» requirement, functional
A statement of some <u>function</u> or <u>feature</u> that should be implemented in a system *[Sommerville 2011]*.

Functional (A) of non-functional (B) ?

The system sends an email to the customer when she places a new order.

functional

- **requirement, non-functional**
A statement of a <u>constraint</u> <…> that applies to a system *[Sommerville 2011]*.



Type: functional vs. non-functional

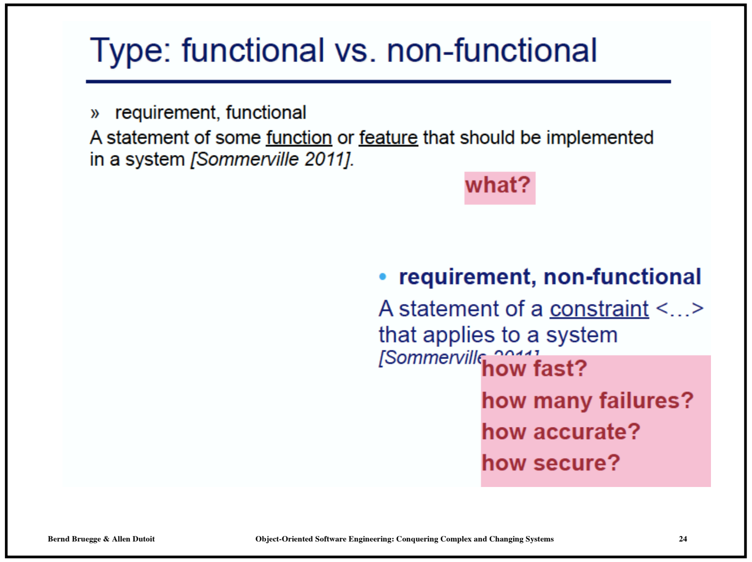
» requirement, functional
A statement of some <u>function</u> or <u>feature</u> that should be implemented in a system *[Sommerville 2011]*.

Functional (A) of non-functional (B)?

The mail should be sent not later than 12 hours after the order has been placed.

non-functional

- **requirement, non-functional**
A statement of a <u>constraint</u> <…> that applies to a system *[Sommerville 2011]*.



Type: functional vs. non-functional

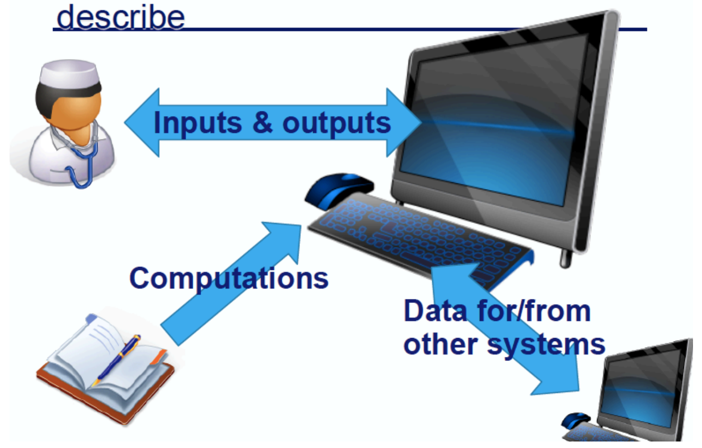
» requirement, functional
A statement of some <u>function</u> or <u>feature</u> that should be implemented in a system *[Sommerville 2011]*.

what?

- **requirement, non-functional**
A statement of a <u>constraint</u> <…> that applies to a system *[Sommerville*

how fast?
how many failures?
how accurate?
how secure?

Functional requirements frequently describe



Non-functional requirements

» Non-functional requirement relates to quality attributes: e.g., **performance, learnability, availability**

» functional requirement: *"when the user presses the green button the Options dialog appears"*:
  – performance: how quickly the dialog appears;
  – availability: how often this function may fail, and how quickly it should be repaired;
  – learnability: how easy it is to learn this function.

Popular Quality Attributes (1)

» reliability
  – availability, fault tolerance, recoverability, ...

» performance
  – time, resource utilization

» operability
  – appropriateness recognizability, ease of use, use of interface aesthetics, technical accessibility, …

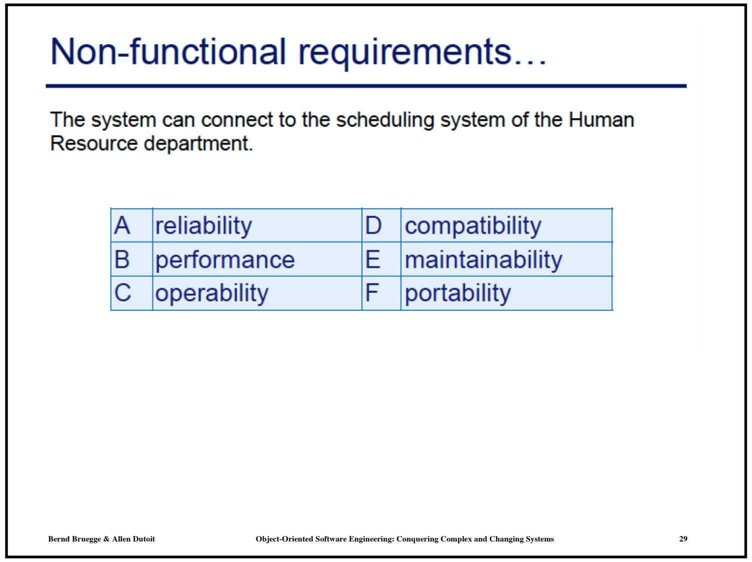
Popular Quality Attributes (2)

» security
  – confidentiality, integrity, authenticity, …

» compatibility
  – co-existence, interoperability

» maintainability
  – modularity, reusability, modifiability, testability, analyzability

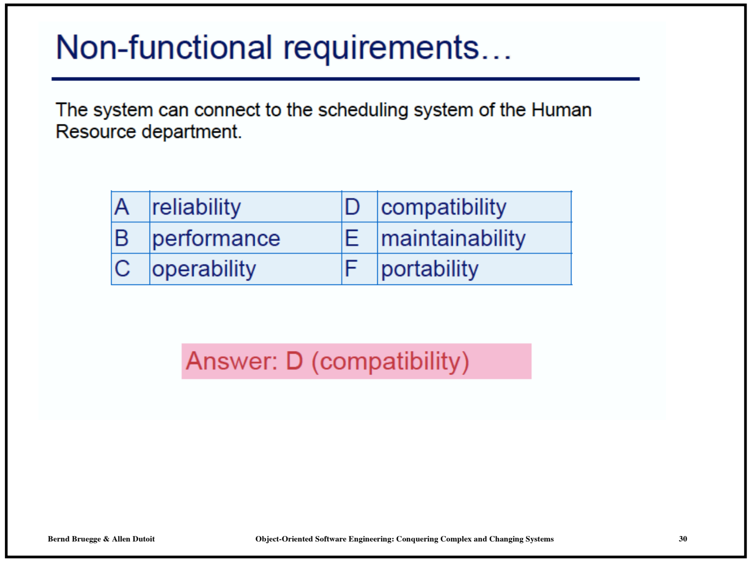
» portability
  – adaptability, replaceability, installability



Non-functional requirements…

The system can connect to the scheduling system of the Human Resource department.

A	reliability	D	compatibility
B	performance	E	maintainability
C	operability	F	portability

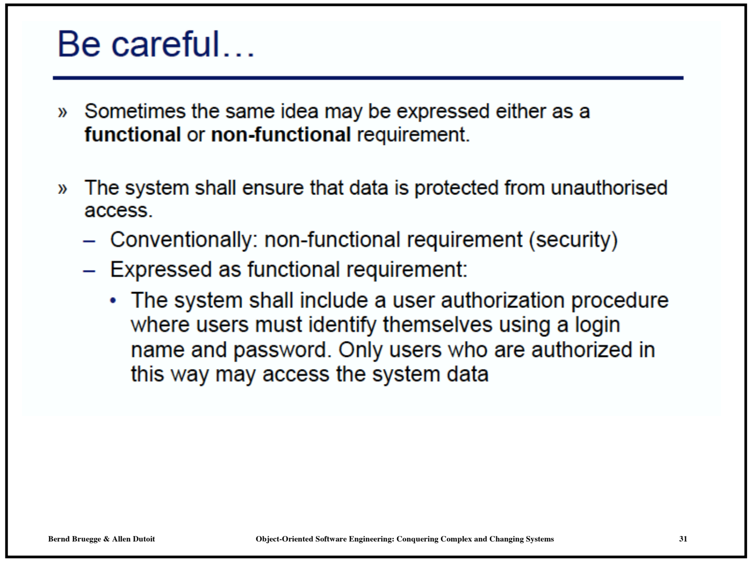


Non-functional requirements…

The system can connect to the scheduling system of the Human Resource department.

A	reliability	D	compatibility
B	performance	E	maintainability
C	operability	F	portability

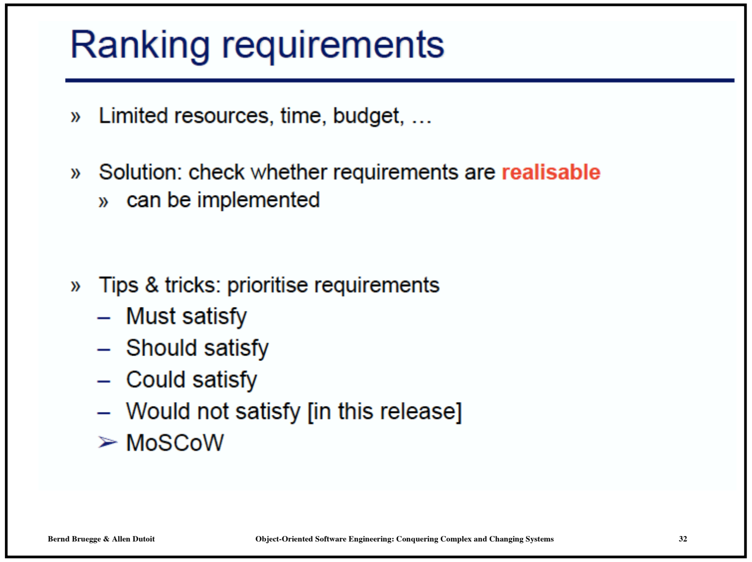
Answer: D (compatibility)



Be careful…

» Sometimes the same idea may be expressed either as a **functional** or **non-functional** requirement.

» The system shall ensure that data is protected from unauthorised access.
- Conventionally: non-functional requirement (security)
- Expressed as functional requirement:
  - The system shall include a user authorization procedure where users must identify themselves using a login name and password. Only users who are authorized in this way may access the system data



Ranking requirements

» Limited resources, time, budget, …

» Solution: check whether requirements are **realisable**
  » can be implemented

» Tips & tricks: prioritise requirements
- Must satisfy
- Should satisfy
- Could satisfy
- Would not satisfy [in this release]

➢ MoSCoW

Requirements attributes

- To support requirements analysis, well-formed requirements should have descriptive attributes defined to help in understanding and managing the requirements. The attribute information should be associated with the requirements in the selected requirements repository.

Examples of requirements attributes

- Identification
  - **Each requirement should be uniquely identified (i.e., number, name tag, mnemonic).**
  - **Identification can reflect linkages and relationships, if needed, or they can be separate from identification.**
  - **Unique identifiers aid in requirements tracing. Once assigned, the identification has to be unique - it is never changed (even if the identified requirement changes) nor is it reused (even if the identified requirement is deleted).**
- Stakeholder Priority.
  - **This may be established through a consensus process among potential stakeholders.**
  - **a scale such as 1-5 or High, Medium, or Low,**

Examples of requirements attributes

- Dependency.
  - **The dependency between requirements should be defined, when a dependency exists.**
  - **Some requirements could have a low priority from one of the stakeholders' perspective, but nevertheless be essential for the success of the system.**
    - **For example, a requirement to measure external ambient temperature could be essential to provide support to other requirements such as the maintenance of internal cabin temperature. This relationship should be identified so that if the primary requirement is removed, the supporting requirement can also be eliminated.**
- Risk.
  - **Major risks are related to potential financial loss, potential missed business opportunity, loss of confidence by stakeholders, environmental impact, safety and health issues, and national standards or laws.**

Examples of requirements attributes

- Source.
  - **Each requirement should include an attribute that indicates the originator. Multiple sources may be considered creators of each requirement. Identifying the sources for each requirement support identifying which organizations(s) to consult for requirement clarification, deconfliction, modification, or deletion. The concept of ownership is related to source. Ownership applies to the origin of a requirement.**
  - **The requirement source indicates where the requirement came from. For stakeholder requirements, the stakeholder who issues the requirement gains ownership.**
- Rationale.
  - **The rationale provides the reason that the requirement is needed and points to any supporting analysis, trade study, modelling, simulation, or other substantive objective evidence.**

Examples of requirements attributes

- Difficulty.
  - **The assumed difficulty for each requirement should be noted (e.g., Easy / Nominal / Difficult).**
  - **This provides additional context in terms of requirements breadth and affordability. It also helps with cost modelling.**
- Type.
  - **Requirements vary in intent and in the kinds of properties they represent. This aids collecting requirements into groups for analysis and allocation.**
  - **Functional, Non-Functional, Quality**

FURPS+: nonfunctional requirements categories

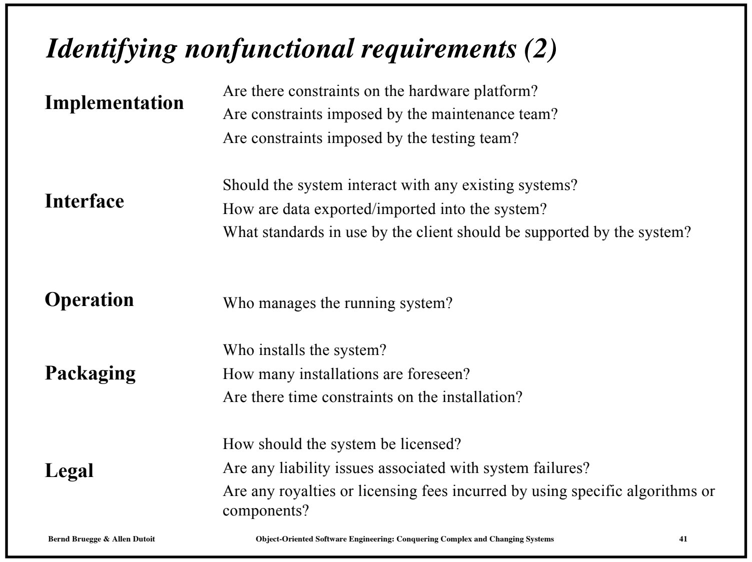
- **Usability** is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.
  - **Conventions adopted by the user interface, the scope of online help, and the level of user documentation, …**
- **Reliability** is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.
  - **Mean time to failure, ability to detect specified faults or to withstand specified security attacks, …**
  - **Recently replaced by** *dependability*, **which is the property of a computer system such that reliance can justifiably be placed on the service it delivers. Dependability includes reliability,** *robustness*, **and** *safety*
- **Performance** requirements are concerned with quantifiable attributes of the system, such as **response time**, **throughput**, **availability**, and **accuracy.**
- **Supportability** requirements are concerned with the ease of changes to the system after deployment, including for example, **adaptability**, **maintainability**, and internationalization.
  - **The ISO 9126 standard on software quality replaces this category with two categories:** *maintainability* **and** *portability*

FURPS+: Pseudo requirements categories

- ***Implementation requirements*** are constraints on the implementation of the system, including the use of specific tools, programming languages, or hardware platforms.
- ***Interface requirements*** are constraints imposed by external systems, including legacy systems and interchange formats.
- ***Operations requirements*** are constraints on the administration and management of the system in the operational setting.
- ***Packaging requirements*** are constraints on the actual delivery of the system (e.g., constraints on the installation media for setting up the software).
- ***Legal requirements*** are concerned with licensing, regulation, and certification issues.

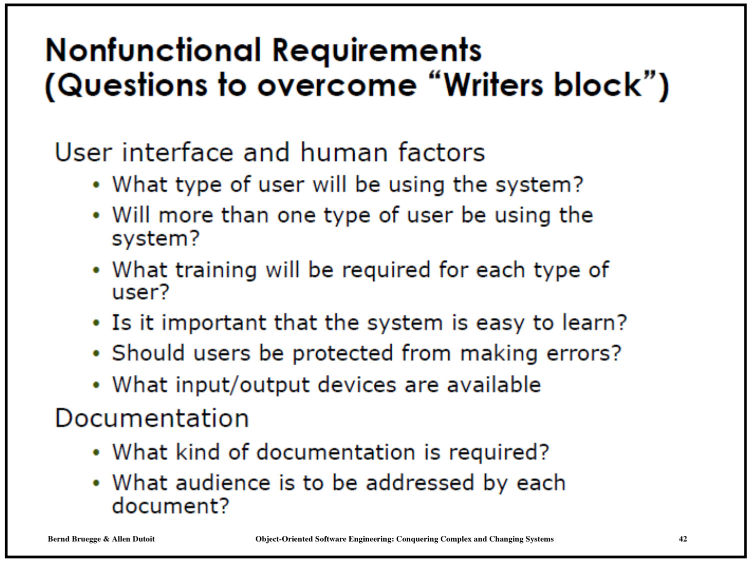
Identifying nonfunctional requirements (1)

Usability	What is the level of expertise of the user? What user interface standards are familiar to the user? What documentation should be provided to the user?
Reliability *(including robustness, safety, and security)*	How reliable, available, and robust should the system be? Is restarting the system acceptable in the event of a failure? How much data can the system loose? How should the system handle exceptions? Are there safety requirements of the system? Are there security requirements of the system ?
Performance	How responsive should the system be? Are any user tasks time critical? How many concurrent users should it support? How large is a typical data store for comparable systems? What is the worse latency that is acceptable to users?
Supportability *(including maintainability and portability)*	What are the foreseen extensions to the system? Who maintains the system? Are there plans to port the system to different software or hardware environments?



Identifying nonfunctional requirements (2)

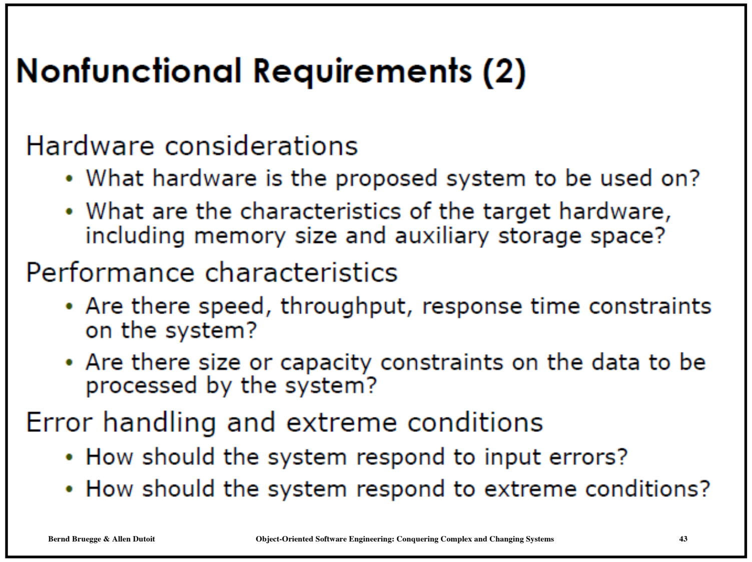
Implementation	Are there constraints on the hardware platform? Are constraints imposed by the maintenance team? Are constraints imposed by the testing team?
Interface	Should the system interact with any existing systems? How are data exported/imported into the system? What standards in use by the client should be supported by the system?
Operation	Who manages the running system?
Packaging	Who installs the system? How many installations are foreseen? Are there time constraints on the installation?
Legal	How should the system be licensed? Are any liability issues associated with system failures? Are any royalties or licensing fees incurred by using specific algorithms or components?



Nonfunctional Requirements (Questions to overcome "Writers block")

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 training will be required for each type of user?
- Is it important that the system is easy to learn?
- Should users be protected from making errors?
- What input/output devices are available

Documentation
- What kind of documentation is required?
- What audience is to be addressed by each document?

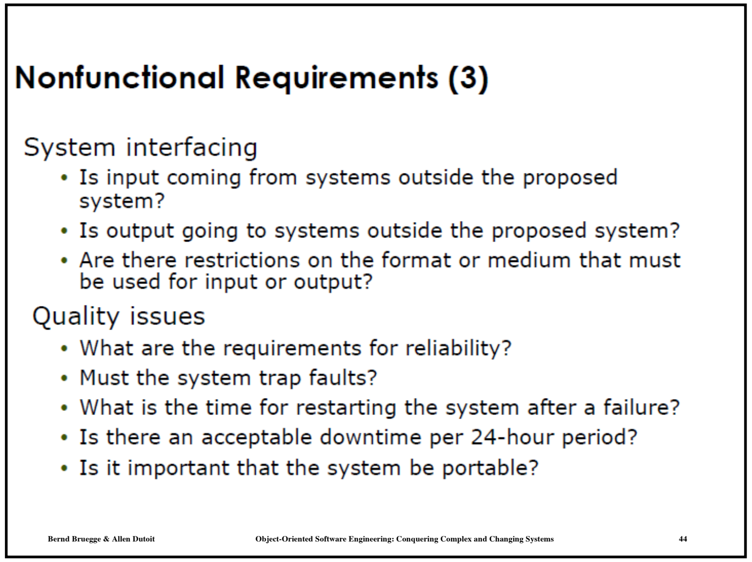


Nonfunctional Requirements (2)

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?

Performance characteristics
- Are there speed, throughput, response time constraints on the system?
- Are there size or capacity constraints on the data to be processed by the system?

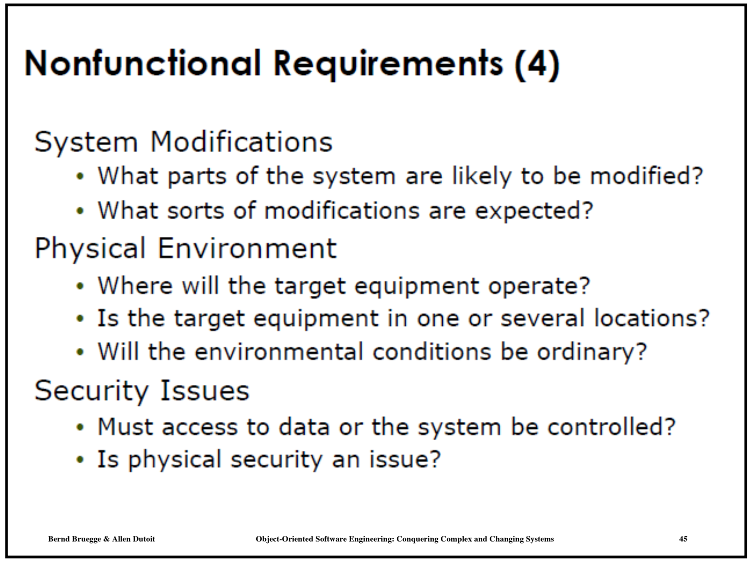
Error handling and extreme conditions
- How should the system respond to input errors?
- How should the system respond to extreme conditions?



Nonfunctional Requirements (3)

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?

Quality issues
- What are the requirements for reliability?
- Must the system trap faults?
- What is the time for restarting the system after a failure?
- Is there an acceptable downtime per 24-hour period?
- Is it important that the system be portable?

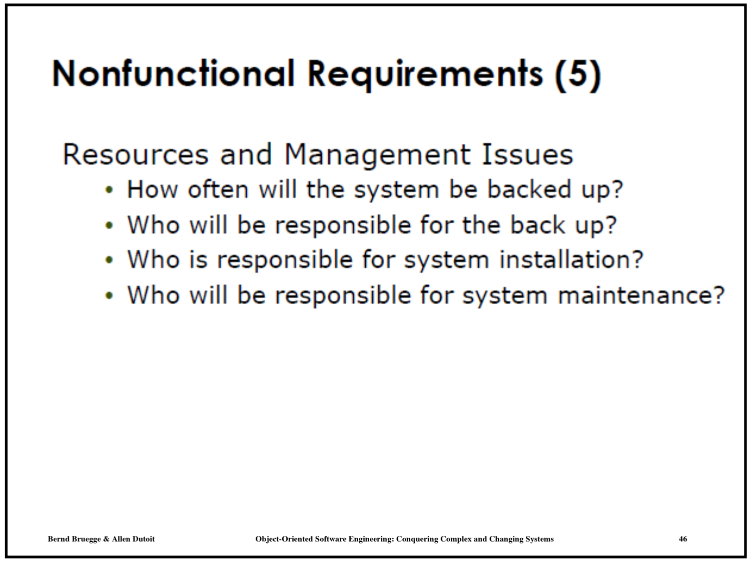


Nonfunctional Requirements (4)

System Modifications
- What parts of the system are likely to be modified?
- What sorts of modifications are expected?

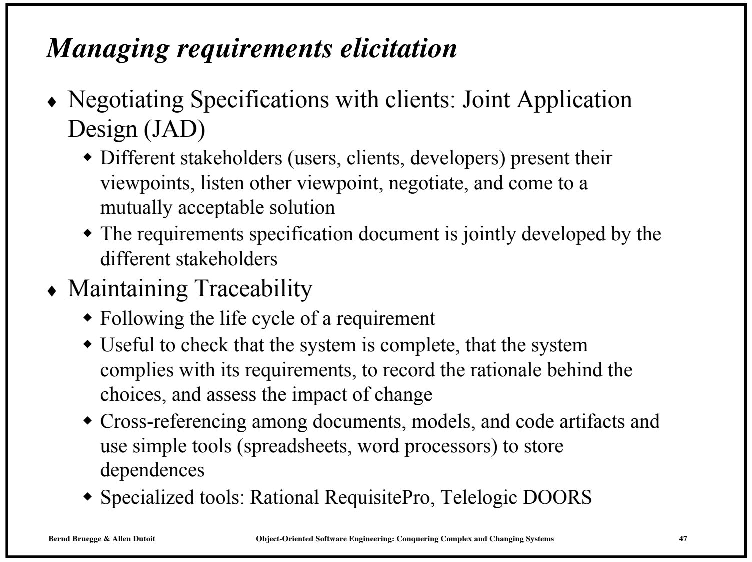
Physical Environment
- Where will the target equipment operate?
- Is the target equipment in one or several locations?
- Will the environmental conditions be ordinary?

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



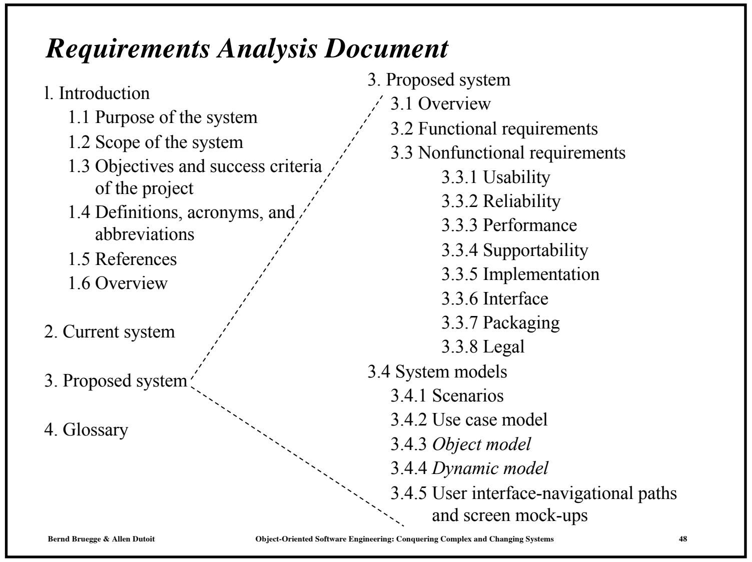
Nonfunctional Requirements (5)

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?



Managing requirements elicitation

- Negotiating Specifications with clients: Joint Application Design (JAD)
  - Different stakeholders (users, clients, developers) present their viewpoints, listen other viewpoint, negotiate, and come to a mutually acceptable solution
  - The requirements specification document is jointly developed by the different stakeholders
- Maintaining Traceability
  - Following the life cycle of a requirement
  - Useful to check that the system is complete, that the system complies with its requirements, to record the rationale behind the choices, and assess the impact of change
  - Cross-referencing among documents, models, and code artifacts and use simple tools (spreadsheets, word processors) to store dependences
  - Specialized tools: Rational RequisitePro, Telelogic DOORS



Requirements Analysis Document

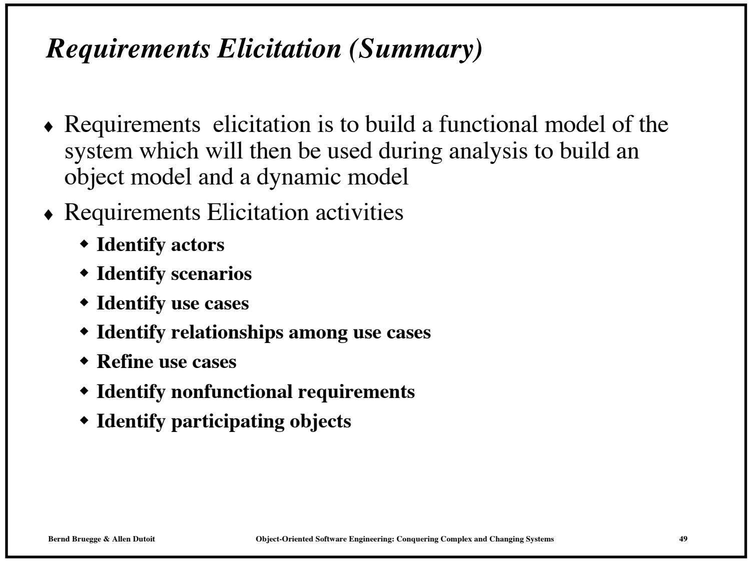
l. Introduction
- 1.1 Purpose of the system
- 1.2 Scope of the system
- 1.3 Objectives and success criteria of the project
- 1.4 Definitions, acronyms, and abbreviations
- 1.5 References
- 1.6 Overview

2. Current system

3. Proposed system

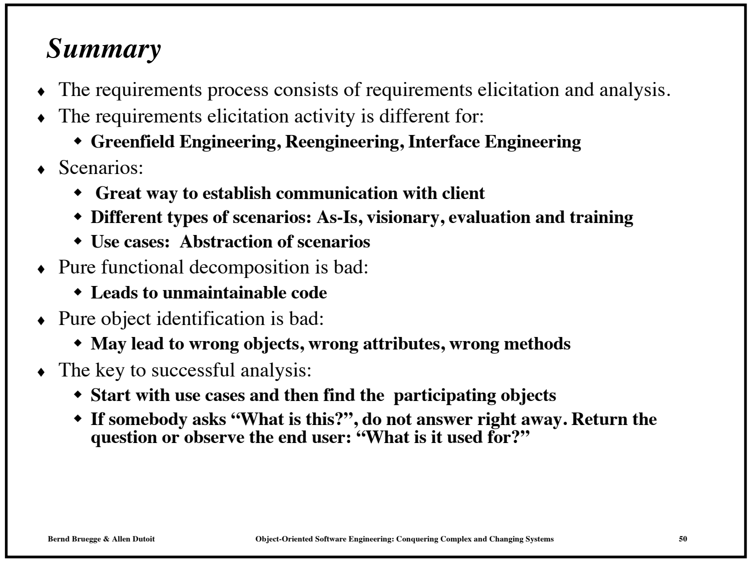
4. Glossary

3. Proposed system
- 3.1 Overview
- 3.2 Functional requirements
- 3.3 Nonfunctional requirements
  - 3.3.1 Usability
  - 3.3.2 Reliability
  - 3.3.3 Performance
  - 3.3.4 Supportability
  - 3.3.5 Implementation
  - 3.3.6 Interface
  - 3.3.7 Packaging
  - 3.3.8 Legal
- 3.4 System models
  - 3.4.1 Scenarios
  - 3.4.2 Use case model
  - 3.4.3 *Object model*
  - 3.4.4 *Dynamic model*
  - 3.4.5 User interface-navigational paths and screen mock-ups



Requirements Elicitation (Summary)

- Requirements elicitation is to build a functional model of the system which will then be used during analysis to build an object model and a dynamic model
- Requirements Elicitation activities
  - **Identify actors**
  - **Identify scenarios**
  - **Identify use cases**
  - **Identify relationships among use cases**
  - **Refine use cases**
  - **Identify nonfunctional requirements**
  - **Identify participating objects**



Summary

- The requirements process consists of requirements elicitation and analysis.
- The requirements elicitation activity is different for:
  - **Greenfield Engineering, Reengineering, Interface Engineering**
- Scenarios:
  - **Great way to establish communication with client**
  - **Different types of scenarios: As-Is, visionary, evaluation and training**
  - **Use cases: Abstraction of scenarios**
- Pure functional decomposition is bad:
  - **Leads to unmaintainable code**
- Pure object identification is bad:
  - **May lead to wrong objects, wrong attributes, wrong methods**
- The key to successful analysis:
  - **Start with use cases and then find the participating objects**
  - **If somebody asks "What is this?", do not answer right away. Return the question or observe the end user: "What is it used for?"**