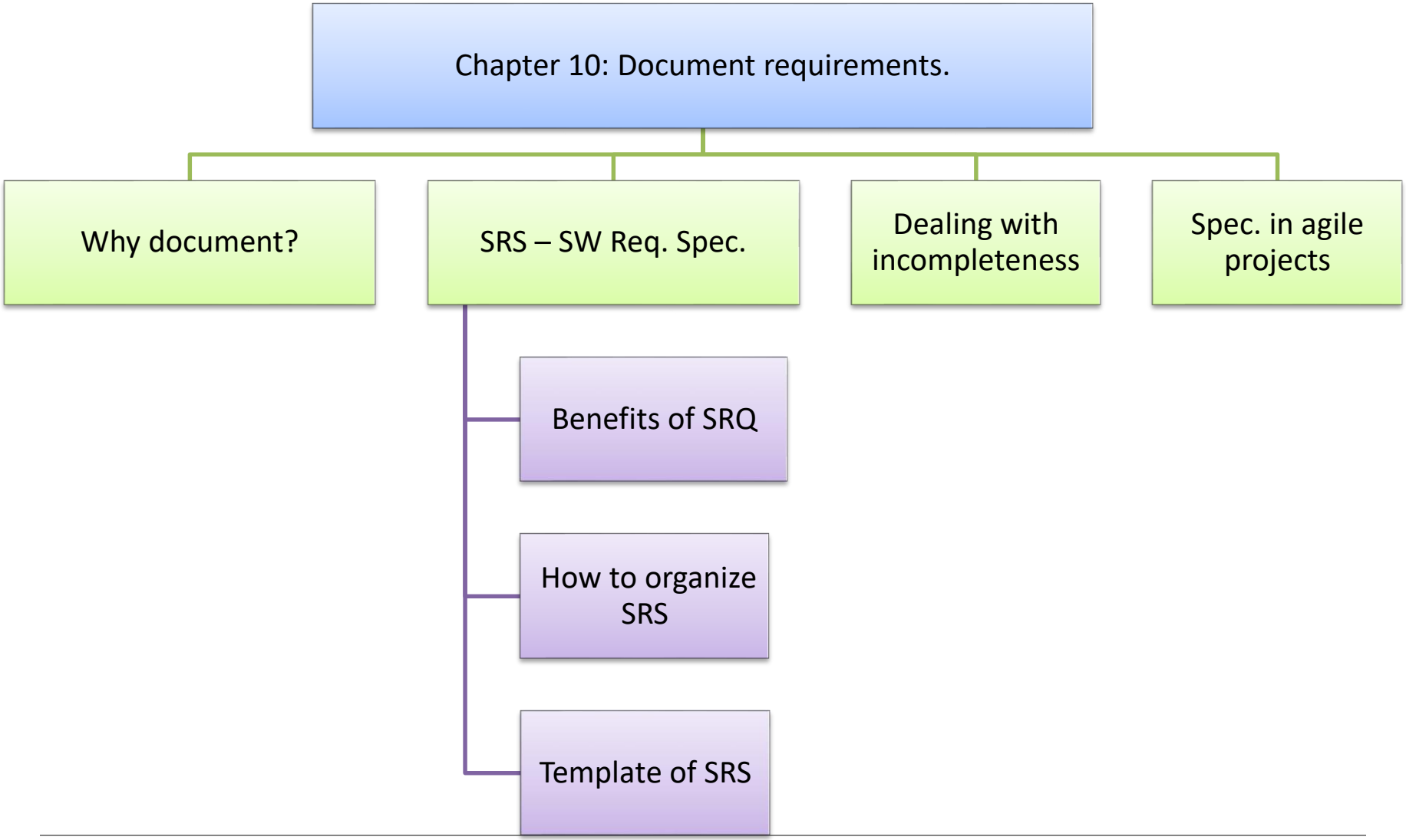
FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

CHAPTER 11

Writing excellent requirements

Review chapter 10



Objectives

- Student could determine if more or less detail is needed in specific areas and how best to represent those requirements.
- Student could examine a page of functional requirements from requirements of student's project to see whether each statement exhibits the characteristics of excellent requirements.
- Student could rewrite any requirements that don't measure up.
- Student could look for conflicts between different requirements in the specification, for missing requirements, and for missing sections of the SRS.

Contents

1. Characteristics of excellent requirements
2. Guidelines for writing requirements
3. Sample requirements, before and after

Characteristics of excellent requirements

- Characteristics of requirement statements
  - Complete: Each requirement must contain all the info necessary for reader to understand it.
  - Correct: Each requirement must accurately describe a capability that will meet some stakeholder's needs and must clearly describe the functionality to be built.
  - Feasible: It must be possible to implement each requirement within the known capabilities and limitations of the system and its operating environment, as well as within project constraints of time, budget, and staff.
  - Necessary: Each requirement should describes a capability that provides stakeholders with the anticipated biz value, differentiates the product in the marketplace, or is required for conformance to an external standard, policy, or regulation.
  - Prioritized: Prioritize biz requirements accordance to which are most important to achieving the desired value. Requirements prioritization should be a collaborative activity involving multiple stakeholders perspectives.
  - Unambiguous: That's when different people read the requirement and come up with different to each of them.
  - Verifiable: Can a tester devise tests (đặt KH test) or other verification approaches to determine whether each requirement is properly implemented?

Characteristics of excellent requirements

- Characteristics of requirements collections: Sets of requirements that are grouped into baseline for a specific release or iteration.
  - Complete: No requirement or necessary information should be absent. In practice, you'll never document every single requirement for any system.
  - Consistent: Consistent requirements don't conflict with other requirements of the same type or with higher-level biz, user, or system requirements.
  - Modifiable: You can always rewrite a requirement, but you should maintain a history of changes made to each requirement, especially after they are baselined.
  - Traceable: A traceable requirement can be linked both backward to its origin and forward to derived requirements, design elements, code that implements it, and test that verify its implementation. Traceable requirements are uniquely labeled with persistent identifiers.

Guidelines for writing requirements

- System or user perspective: You can write functional requirements from the perspective (quan điểm) of either something the system does or something the user can do.
- Writing style: Being easy to read and understand is an essential element of well-written requirements
  - Clarity and conciseness: Write requirements in complete sentences using proper grammar, spelling, and punctuation. Keep sentences and paragraphs sort and direct, avoiding jargon. Define specialized terms in a glossary.
  - The keyword “shall”: In general, the BAs often use “shall” in writing requirements. Because author use “shall” to indicate a requirements and “will” to denote (bao hàm) a design expectation.
  - Active voice: Write in the active voice to make it clear what entity is taking the action described.
  - Individual requirements: Avoid writing long narrative paragraphs (đoạn văn dài dòng) that contain multiple requirements. In this case, split the sentence into separate requirements.

EX for individual requirements

"The Buyer's credit card on file shall be charged for payment, unless the credit card has expired."

Chuyển thành 2 requirements đơn:

"If the Buyer's credit card on file is active, the system shall charge the payment to this card."

Và

"If the Buyer's credit card on file has expired, the system shall allow the Buyer to either update the current credit card information or enter a new credit card for payment."

Guidelines for writing requirements

- **Level of detail:** Requirements need to be specified at a level of precision that provides developers and testers with just enough information to properly implement them.
  - **Appropriate detail:** An important part of requirement analysis is to decompose a high-level requirement into sufficient detail to clarify it and flesh it out (giải quyết nó). The fewer the opportunities for ongoing discussion about requirements issues, the more specifics you need to record in the requirements set.
  - **Consistent granularity** (độ chi tiết nhất quán):
    - *EX1: The system shall interpret the keystroke combination Ctrl+S as File Save*
    - *EX2: The system shall interpret the keystroke combination Ctrl+P as File Print*

Guidelines for writing requirements

- Representation techniques: Some alternatives to the natural language requirements that we're used to are lists, tables, visual analysis models, charts, mathematical formulas, photographs, sound clips, and video clips.
  - Sample: *The Text Editor shall able to parse <format> documents that define <jurisdiction> laws.*
  - Table 11-1 Requirements for parsing documents

Jurisdiction	Tagged format	Untagged format	ASCII format
Federal	.1	.2	.3
State	.4	.5	.6
Territorial	.7	#N/A	.8
International	.9	.10	.11

  - EX1: ***Editor.DocFormat*** *The Text Editor shall be able to parse documents in several formats that define laws in jurisdictions (quyền pháp lý) shown in table 11-1.*
  - EX2: ***Editor.DocFormat.3*** *The Text Editor shall be able to parse ASCII documents that define federal laws.*

Guidelines for writing requirements

- Avoiding ambiguity: Requirements quality is in the eye of the reader, not the author! The requirements are written crystal clear, free from ambiguities and other problems.
  - Fuzzy words: Use terms consistently and as defined in the glossary. Watch out (coi chừng) for synonyms and near-synonyms. Avoid words like *reasonably, appropriately, generally, usually, friendly, quickly,….* *Read page 214 in your textbook.*
  - The A/B construct:
    - EX: *The system shall provide automated information collection of license key data for a mass release from the Delivery/Fulfillment Team.*
    - *Explanation: The name of team is Delivery/Fulfillment; Delivery and Fulfillment are synonyms; Some projects call Delivery Team, and other call Fulfillment Team;…*
  - Boundary values:
    - *EX: Vacation request of up to 5 days do not require approval. Vacation requests of 5 to 10 days require supervisor approval. Vacation request of 10 days or longer require management approval.*
  - Negative requirements: Requirement that say what the system *will no* do rather than *will* do. How do you implement a don't-do-this requirement?
    - *EX: Prevent the user from activating the contract if the contract is not in balance.*
    - → *The system shall allow the user to activate the contract only if the contract is in balance.*

Guidelines for writing requirements

- Avoiding incompleteness
  - Symmetry (đối xứng): Symmetrical operations are a common source of missing requirements.
    - *EX: The user must be able to save the contract at any point during manual contract setup.*
    - *Explanation:* The system can't validate the data entries in the incomplete contract.
  - Complex logic: Compound logical expressions often leave certain combinations of decision values undefined.
    - *EX: If the Premium plan is not selected and proof of insurance is not provided, the customer should automatically default into the Basic plan.*
    - *Explanation: There are 3 possibilities:* The Premium plan is selected and proof of insurance is not provided; The Premium plan is selected and proof of insurance is provided; The Premium plan is not selected and proof of insurance is provided;

    → In this case it's better to using decision tree or decision tables.
  - Missing exceptions:
    - *EX: If the user is working in an existing file and chooses to save the file, the system shall save it with the same name.*
    - *Explanation:* This requirement doesn't indicate what the system do if it's unable to save the file with the same name.

    → *If the system is unable to save a file using the same name, the system shall give the user the option to save it with a different name or to cancel the save operation.*

Sample requirements, before and after

- The groups read carefully 3 examples from page 218 to page 219 for more information about the writing requirements' skills.

- **Example 1** *The Background Task Manager shall provide status messages at regular intervals not less than every 60 seconds.*
- **Example 2** *Corporate project charge numbers should be validated online against the master corporate charge number list, if possible.*
- **Example 3** *The device tester shall allow the user to easily connect additional components, including a pulse generator, a voltmeter, a capacitance meter, and custom probe cards.*

Teamwork

- Read carefully the following requirement:

"The system must check for inconsistencies in account data between the Active Account Log and the Account Manager archive. The logic that is used to generate these comparisons should be based on the logic in the existing consistency checker tool. In other words, the new code does not need to be developed from scratch. The developers should utilize the current consistency checker code as the foundation. However, additional logic must be added to identify which database is the authoritative source. The new functionality will include writing data to holding tables to indicate how/where to resolve inconsistencies. Additionally, the code should also check for exception scenarios against the security tools database. Automated email alerts should be sent to the Security Compliance Team whenever discrepancies are found."

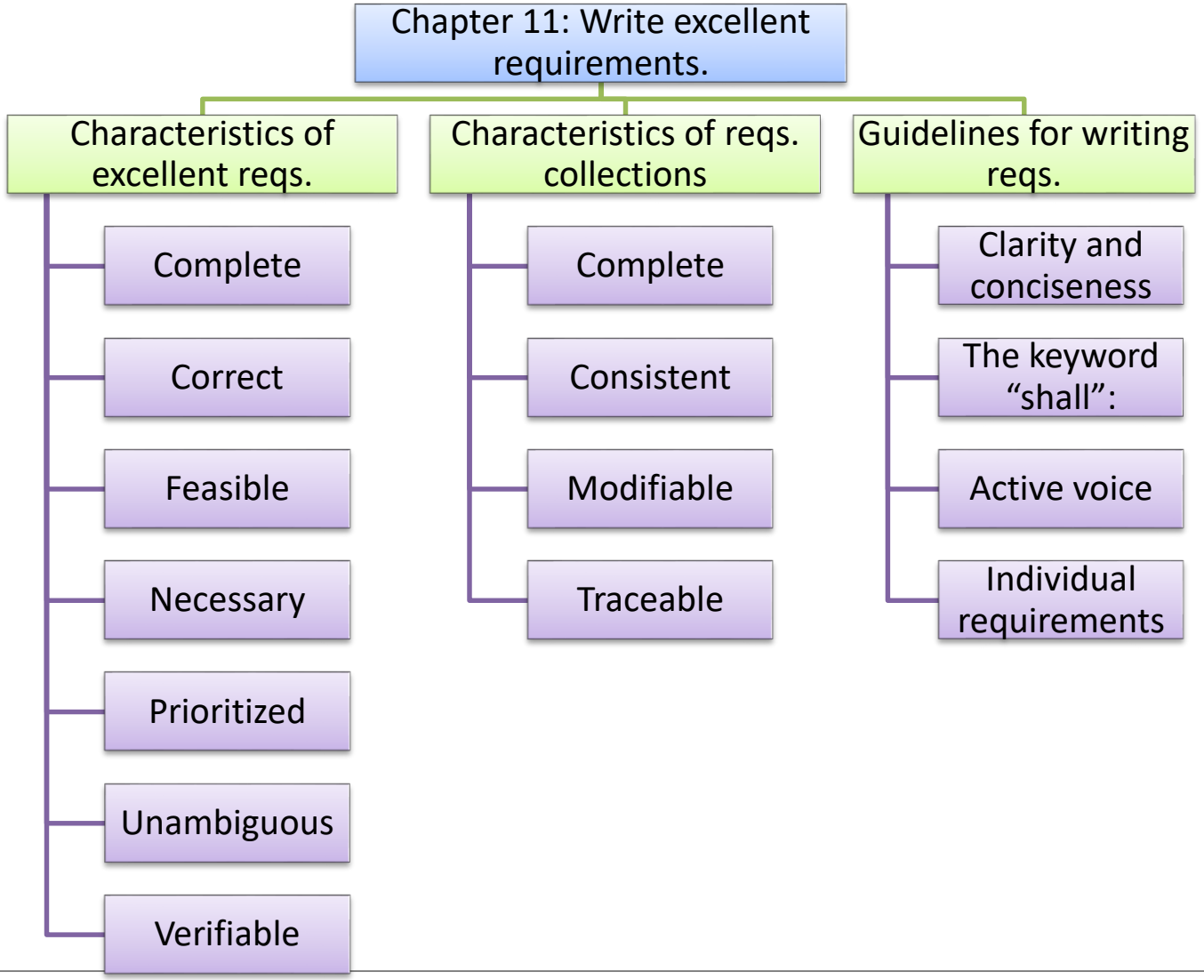
Teamwork

- Read carefully the following guidelines:
  - There are numerous requirements in here that should be split out individually.
  - If the comparison logic is “based on” logic in the existing consistency checker tool, exactly what portion of the code can be reused and how does it need to be changed? What functions are different between the new system and the existing tool? What “additional logic” must be added? How exactly can the system determine “which database is the authoritative source”?
  - The new functionality “includes” writing data to holding tables; is that all, or is other functionality “included” that isn’t explicitly stated?
  - Clarify what “how/where” means when resolving inconsistencies.
  - “Should” is used in several places.
  - What’s the relationship between an “exception scenario” and a “discrepancy”? If they’re synonyms, pick one term and stick with it. A glossary might clarify whether these are the same or how they are related.
  - What information should the system send to the Security Compliance Team when it detects a discrepancy?

Teamwork

- Your group's tasks:
  - Rewrite the above requirements.
  - Save in the document (.doc).
  - Submit it via my email.
  - Deadline: 20:00 PM tomorrow.

Review chapter 11



THE END
THANK YOU!