SE202
Introduction to Software Engineering

Lecture 4-3
Software Requirement Specification (SRS)

Pathathai Na Lumpoon

Last class

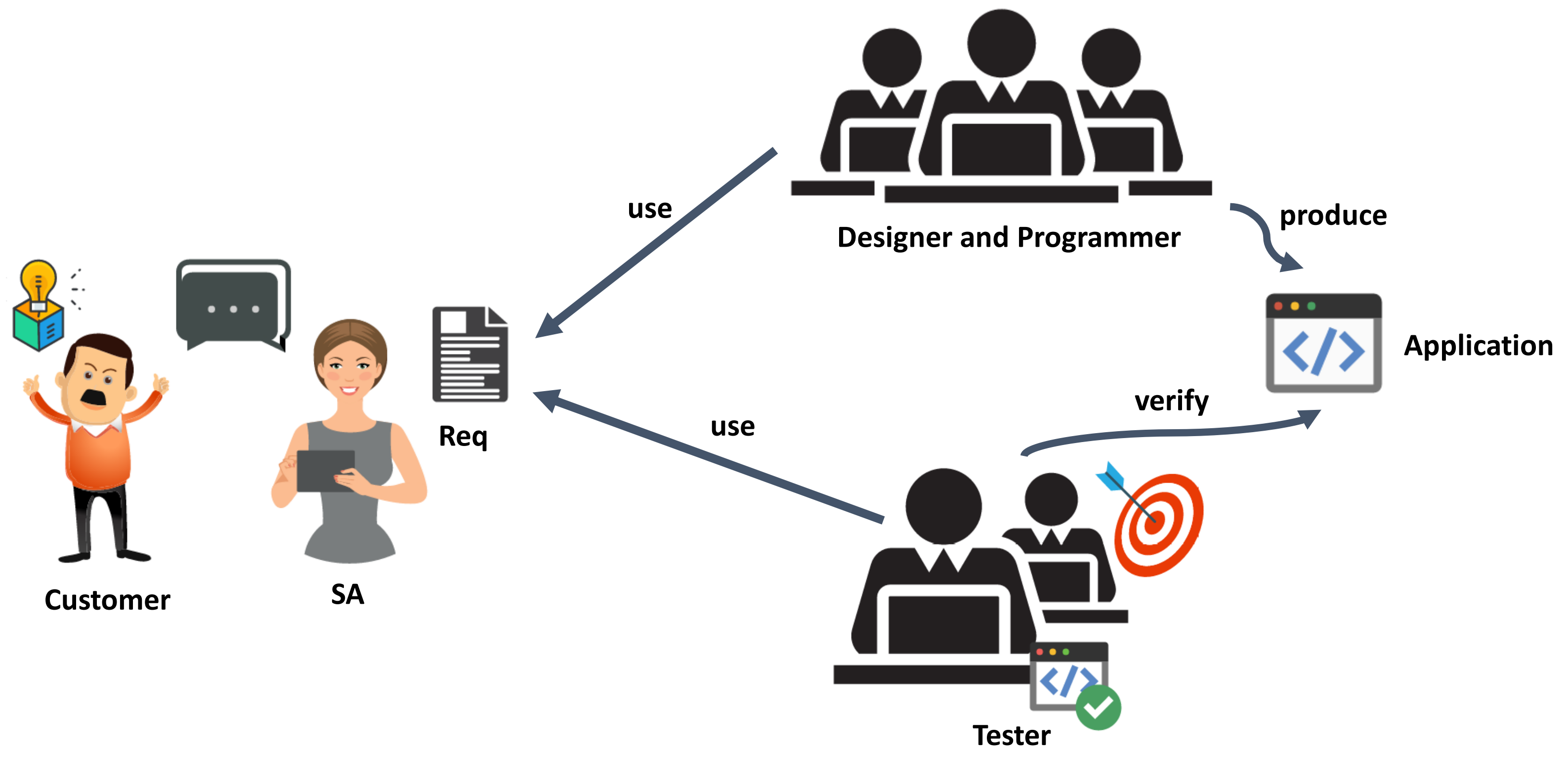
- Use case

Today

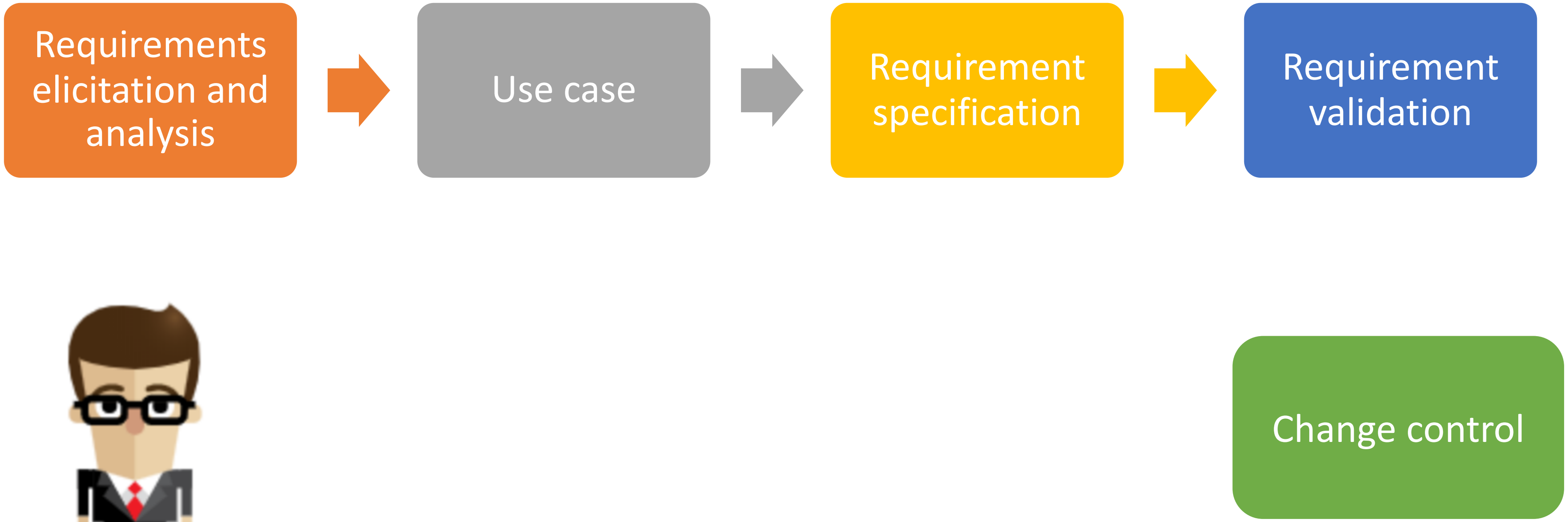
- Software Requirement Specification (SRS)

What is Requirement?

- Requirements are the features that your application must be developed.
- All later work products—software design and architecture, code, test plans—are based on the requirements



Softwre Requirements Engineering


Requirements elicitation and analysis
Use case
Requirement specification
Requirement validation
Change control

SA or BA

The goal of the requirement is to communicate this behavior in as clear and unambiguous a manner as possible.

Software requirements specification (SRS)

- A complete description of the behavior of the software to be developed
- Two types of requirement in SRS
    - Functional requirements
        - The internal workings of the software
        - E.g. the calculations, technical details, data manipulation and processing, and other specific functionality
    - Nonfunctional requirements
        - The constraints on the design or implementation
        - E.g. quality requirements, platform requirements, or process requirements

Functional requirements

- What inputs the system should accept
- What outputs the system should produce
- What data the system should store that other systems might use
- What computations the system should perform
- The timing and synchronization of the above

Nonfunctional requirements

- Users have implicit expectations about how well the software will work
  - How easy the software is to use, how quickly it executes, how reliable it is and how well it behaves when unexpected conditions arise.
- The nonfunctional requirements should be defined as precisely as possible (quantifying them)
  - E.g.
  - The maximum number of seconds it must take to perform a task (Response time)
  - The maximum size of a database on disk (Resource usage)
  - The number of hours per day a system must be available (Availability)

Nonfunctional requirements - Quality requirements

- Categories reflecting **usability, efficiency, reliability, maintainability and reusability**
    - Response time
    - Throughput
    - Resource usage
    - Reliability
    - Availability
    - Recovery from failure
    - Allowances for maintainability and enhancement
    - Allowances for reusability
- All must be verifiable

Nonfunctional requirements - Platform requirements

- Categories constraining the environment and technology of the system.
  - Platform
  - Technology to be used

Nonfunctional requirements - Process requirements

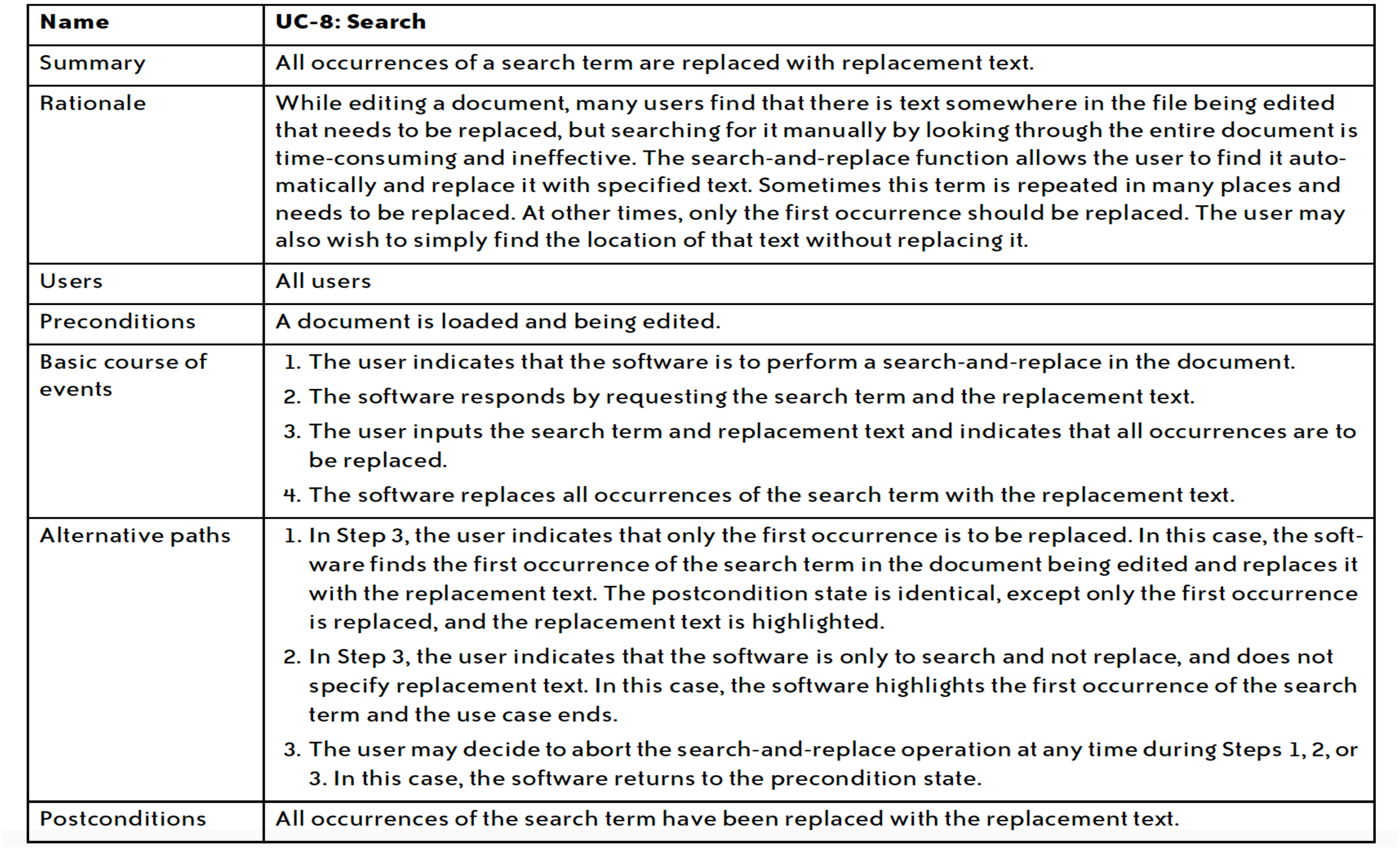
- Categories constraining the project plan and development methods
  - Development process (methodology) to be used
  - Cost and delivery date
  - Often put in contract or project plan instead

Questions on Google classroom

SRS Template

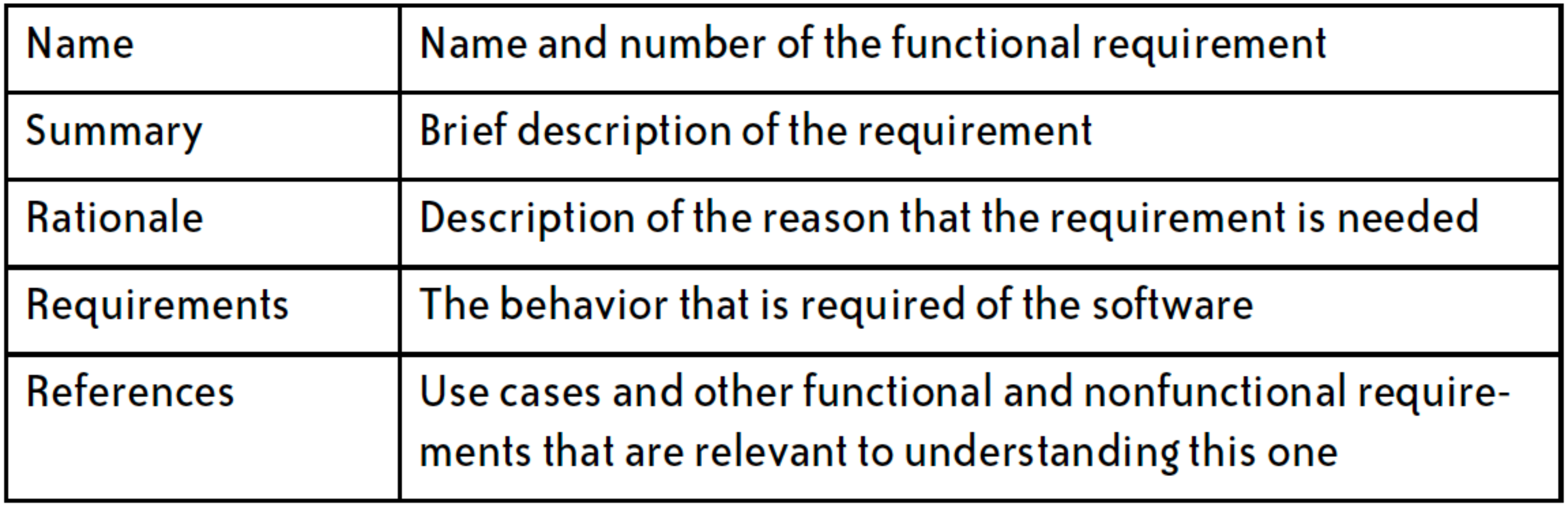
1. Introduction
    1. Purpose of the document
    2. Scope of the document
    3. System overview (a brief summary of the vision and scope of the project)
    4. References
2. Definitions (glossary terms that the reader may not be familiar with)
3. Use cases
4. Functional requirements
5. Nonfunctional requirements

Use cases - List of use case tables



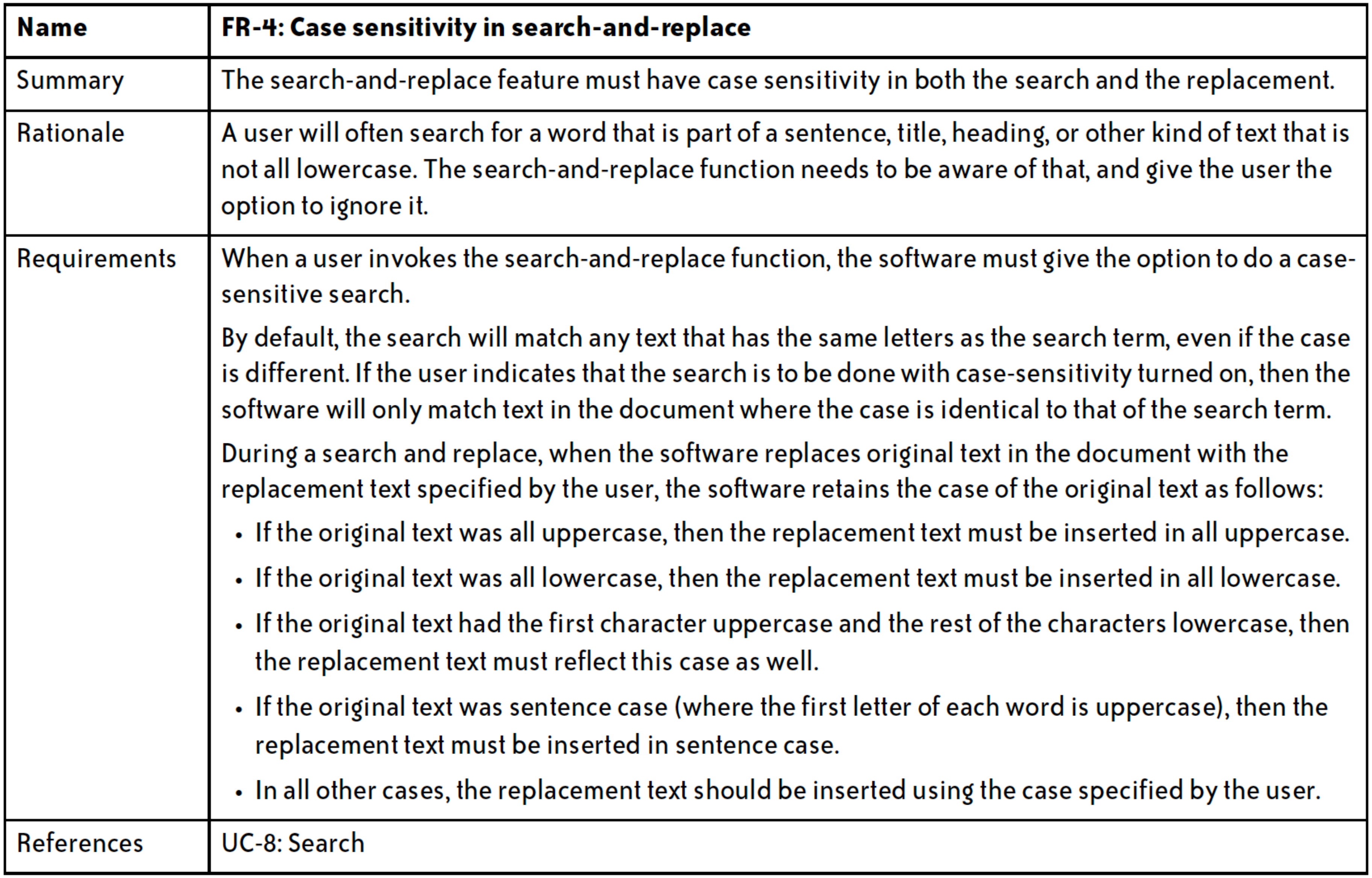
Name	**UC-8: Search**
Summary	All occurrences of a search term are replaced with replacement text.
Rationale	While editing a document, many users find that there is text somewhere in the file being edited that needs to be replaced, but searching for it manually by looking through the entire document is time-consuming and ineffective. The search-and-replace function allows the user to find it automatically and replace it with specified text. Sometimes this term is repeated in many places and needs to be replaced. At other times, only the first occurrence should be replaced. The user may also wish to simply find the location of that text without replacing it.
Users	All users
Preconditions	A document is loaded and being edited.
Basic course of events	1. The user indicates that the software is to perform a search-and-replace in the document. 2. The software responds by requesting the search term and the replacement text. 3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced. 4. The software replaces all occurrences of the search term with the replacement text.
Alternative paths	1. In Step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted. 2. In Step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends. 3. The user may decide to abort the search-and-replace operation at any time during Steps 1, 2, or 3. In this case, the software returns to the precondition state.
Postconditions	All occurrences of the search term have been replaced with the replacement text.

Functional and nonfunctional requirement template



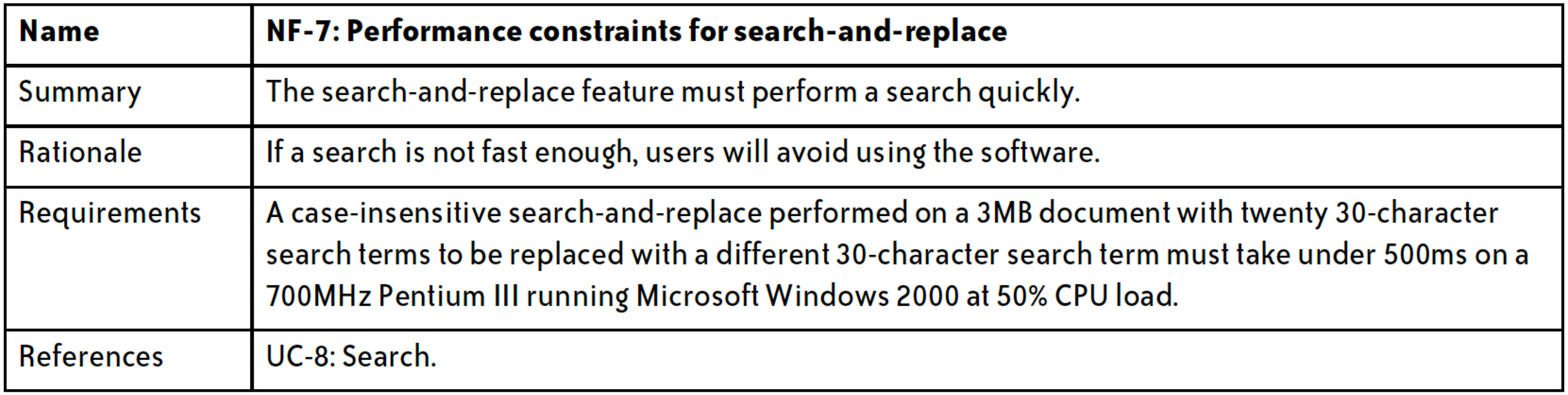
Name	Name and number of the functional requirement
Summary	Brief description of the requirement
Rationale	Description of the reason that the requirement is needed
Requirements	The behavior that is required of the software
References	Use cases and other functional and nonfunctional requirements that are relevant to understanding this one

Functional requirement example



Name	**FR-4: Case sensitivity in search-and-replace**
Summary	The search-and-replace feature must have case sensitivity in both the search and the replacement.
Rationale	A user will often search for a word that is part of a sentence, title, heading, or other kind of text that is not all lowercase. The search-and-replace function needs to be aware of that, and give the user the option to ignore it.
Requirements	When a user invokes the search-and-replace function, the software must give the option to do a case-sensitive search. By default, the search will match any text that has the same letters as the search term, even if the case is different. If the user indicates that the search is to be done with case-sensitivity turned on, then the software will only match text in the document where the case is identical to that of the search term. During a search and replace, when the software replaces original text in the document with the replacement text specified by the user, the software retains the case of the original text as follows: • If the original text was all uppercase, then the replacement text must be inserted in all uppercase. • If the original text was all lowercase, then the replacement text must be inserted in all lowercase. • If the original text had the first character uppercase and the rest of the characters lowercase, then the replacement text must reflect this case as well. • If the original text was sentence case (where the first letter of each word is uppercase), then the replacement text must be inserted in sentence case. • In all other cases, the replacement text should be inserted using the case specified by the user.
References	UC-8: Search

Nonfunctional requirement example



Name	**NF-7: Performance constraints for search-and-replace**
Summary	The search-and-replace feature must perform a search quickly.
Rationale	If a search is not fast enough, users will avoid using the software.
Requirements	A case-insensitive search-and-replace performed on a 3MB document with twenty 30-character search terms to be replaced with a different 30-character search term must take under 500ms on a 700MHz Pentium III running Microsoft Windows 2000 at 50% CPU load.
References	UC-8: Search.

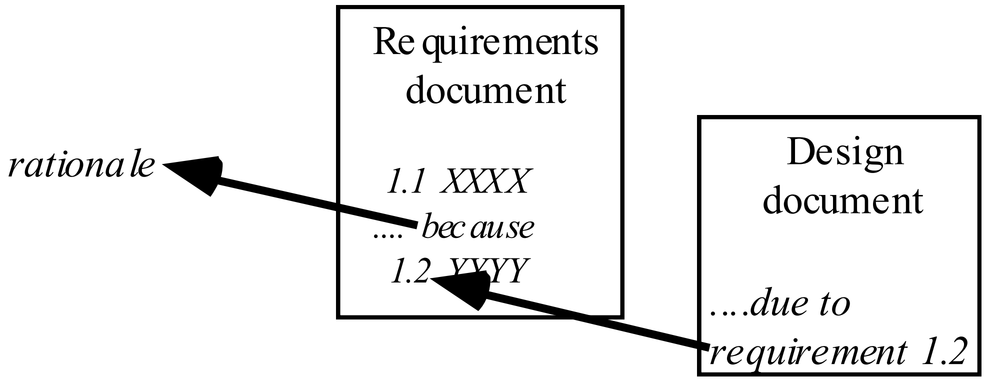
Requirement Validation

- The goal of the SRS development script is to remove as many defects as possible from the SRS.
- A defect is any planned software behavior a project team member, user, stakeholder, or decision-maker does not agree with.
    - E.g. Somebody does not believe that the planned behavior will satisfy the users' or stakeholders' needs.
    - An inspector does not understand what's written or feels that it is ambiguous or confusing.
- The best way to prevent defects is through iteration

Requirements documents...

- The document should be:
  - sufficiently complete
  - well organized
  - clear
  - agreed to by all the stakeholders

- Traceability:



Reviewing Requirements

- **Each individual requirement should**
  - Have **benefits that outweigh the costs** of development
  - Be **important** for the solution of the current problem
  - Be expressed using a **clear and consistent notation**
  - Be **unambiguous**
  - Be **logically consistent**
  - Lead to a system of **sufficient quality**
  - Be **realistic** with available resources
  - Be **verifiable**
  - Be uniquely **identifiable**
  - **Does not over-constrain the design** of the system

Words to Avoid ⊗

- **Comparatives**
  - Words like faster, better, more, and shinier. How much faster? Define “better.” How much more? These need to be quantified.
- **Imprecise adjectives**
  - Words like fast, robust, user-friendly, efficient, flexible, and glorious. These are just other forms of the comparatives.
- **Vague commands**
  - Words like minimize, maximize, improve, and optimize. Unless you use these in a technical algorithmic sense.

Avoid Overly Specific Selections

- Suppose you’re building a phone application
  - Customers place orders at a sandwich and bagel shop called The Loxsmith.
  - The program should let customers select the toppings they want on their bagels.
    - They include lox (naturally), butter, cream cheese, gummy bears, and so on.

Too specific requirement
- The toppings form will display a list of toppings. The user can check boxes next to the toppings to add them to the bagel.

Fixed requirement
- The toppings form will allow the user to select the toppings put on the bagel.

What's wrong with the following statement of functional requirements for a Restaurant Advisor System?

This system will allow people to choose a restaurant in a city. Users enter one or more of the following criteria, and then the system searches its database for suitable restaurants: food type, price range, neighborhood, size, service type (fast food, etc.), smoking arrangements (none, etc.) The user can also specify a desired day and time period, and the number of people in the party. The system will tap into the reservation database and only display restaurants with available space. After entering the criteria, the user clicks on “Search” and the system displays a list of matching restaurants. For restaurants that participate in the reservation system, the user can click on “Reserve” next to a selection to make the reservation.

Prioritized

- With the limit of time and budget, it’s likely you’ll need to cut a few nice-to-haves from the design when you start working on the project’s schedule.
- Customers sometimes have trouble deciding which requirements they can live without.

THE MOSCOW METHOD

- MOSCOW is an acronym to help you remember a common system for prioritizing application features.
- **M—Must.**
  - These are required features that must be included. They are necessary for the project to be considered a success.
- **S—Should.**
  - These are important features that should be included if possible. If there’s a work-around and there’s no room in the release 1 schedule, these may be deferred until release 2.
- **C—Could.**
  - These are desirable features that can be omitted if they won’t fit in the schedule. They can be pushed back into release 2, but they’re not as important as the “should” features, so they may not make it into release 2, either.
- **W—Won’t.**
  - These are completely optional features that the customers have agreed l will not be included in the current release. They may be included in a future release if time permits.

Example: ClassyDraw (a fictional drawing program)

1. Draw: line segments, sequences of line segments, splines, polygons, ellipses, circles, rectangles, rounded rectangles, stars, images, and other shapes.
2. Save and load files
3. Protect the current drawing. For example, if the user tries to close the program while there are unsaved changes, prompt the user.
4. Let the user specify the line style and colors used to draw shapes.
5. L et the user specify the fill style and colors used to draw shapes.
6. Click to select an object.
7. Click and drag to select multiple objects.
8. 8. Click or click and drag with the Shift key down to add objects to the current selection.
9. Click or click and drag with the Ctrl key down to toggle objects in and out of the current selection.
10. Click and drag the selected objects to move them.
11. Edit the selected objects' line and fill styles. .
12. Delete the selected objects.
13. Select colors from a palette.
14. Place custom colors in a custom palette.
15. Support transparency.
16. Copy and paste the entire drawing, a rectangular selection, or an irregular selection as a bitmapped image.
17. Copy, cut, and paste the currently selected objects.
18. Allow the user to write scripts to add shapes to a drawing.
19. Let the user rearrange the palettes and toolbars.
20. Auto-save the current drawing periodically. If the program crashes, allow the user to reload the most recently saved version.
21. Auto-save the current drawing every time a change is made. If the program crashes, allow the user to reload the most recently saved version.
22. Provide online help.
23. Provide online tutorials.

MOSCOW method to prioritize these requirements of ClassyDraw

Must 1 (partial), 2, 3, 6, 10, and 12.	**Should** 1 (remaining), 4, 5, 7, 8, 9, 13, 22, and 23.
Could 11,14,15,16 and 17	**Won’t** 18, 19, 20, and 21

After you’ve assigned each requirement to a category, go back through them and make sure you’re happy with their assignments. If a requirement in the “could” category seems more important than one in the “should” category, switch them

Managing Changing Requirements

- Requirements change because:
    - Business process changes
    - Technology changes
    - The problem becomes better understood
- Need a change control broad to make decision on the project to evaluate the impact of a change before implementing it.

- Requirements analysis *never stops*
    - Skip the software requirements activities will always come back to damage the project later
    - Continue to interact with the clients and users
    - The benefits of changes must outweigh the costs.
        - Certain *small* changes (e.g. look and feel of the UI) are usually quick and easy to make at relatively little cost.
        - *Larger-scale* changes have to be carefully assessed