# FIT5SE1 Software Engineering 1

Lecture 7(b):
Requirement engineering

# Outline

- What is requirement engineering?
- Types of requirement
- Requirement capturing

# References

- Liskov & Guttag (2001):
  - Chapter 11
- Sommerville (2011):
  - Chapter 4 (4.3)

# What is requirement engineering?

- RE is a process to:
  - capture,
  - analyse,
  - document, and
  - check ***what*** services a software provide
- Iterative:
  - incrementally refine the requirements

# Iterative RE

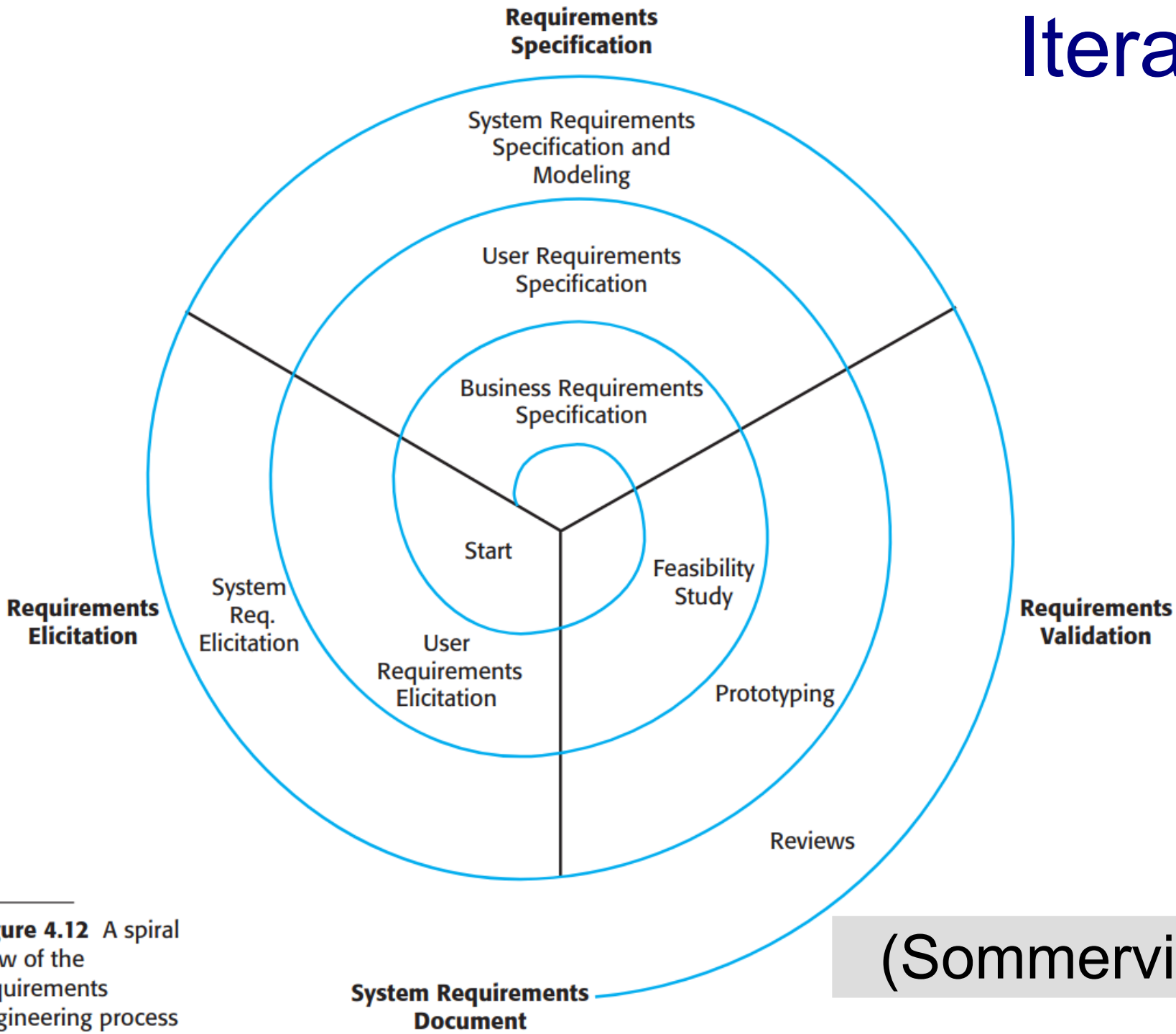



**Figure 4.12** A spiral view of the requirements engineering process

(Sommerville, 2011)

# Types of requirement

- Functional
- Non-functional

# Functional requirements

- Statements about functions and data
- Data: statements about the *entities* of interests
  - written in a structured form:
    - Entity name: <attributes>
    - Relationship name (entities): <attributes>
- Derived from *normal* and *erroneous* user interactions:
  - normal: results in a normal program state
  - erroneous: results in errorneous program state

# KEngine normal FRs

- *the first thing that the user should do is to identify the documents of interest...presenting the URL of a site*

  → ***obtain documents from an URL***

# KEngine functions (1)

| Functions | | Descriptions |
|---|---|---|
| **F1** | Obtain documents | to retrieve web documents from a given URL, which could be the URL of a local folder or of a remote web site |

- *the system run queries against the collection...presenting information about documents containing the keyword*

→ ***search for documents by <u>keyword</u>***

- *the customer requests the ability to "refine" a query by providing another keyword (the matching documents must contain all the keywords)*

  → ***<u>incrementally</u> search for documents by <u>keywords</u>***

# KEngine functions (2)

| Functions | | Descriptions |
|---|---|---|
| **F1** | Obtain documents | to retrieve web documents from a given URL, which could be the URL of a local folder or of a remote web site |
| **F2** | Search for documents | to search the documents collection for the documents that contain the keywords of a query; allowing the user to refine query with more keywords |

- *the user should be able to search the (documents) collection for a document given a title*

  → ***display a document***

# KEngine functions (3)

| Functions | | Descriptions |
|---|---|---|
| **F1** | Obtain documents | to retrieve web documents from a given URL, which could be the URL of a local folder or of a remote web site |
| **F2** | Search for documents | to search the documents collection for the documents that contain the keywords of a query |
| **F3** | Display a document | to retrieve a document from the documents collection given its title |

# KEngine errorneous

- System errors: none
- User errors:

  E1: *Enters a wrong, empty-target, or duplicate URL*

  → informs with an error

  - E2: *Enters a non-keyword*

    → informs with an error

  - E3: *Enters a non-existent word*

    → returns an empty result

# KEngine functions (4)

| Functions | Descriptions |
| --- | --- |
| Obtain documents | E1: If the user enters a wrong, empty-target, or duplicate URL, the system informs with an error |

# KEngine functions (5)

| Functions | Descriptions |
|---|---|
| Obtain documents | E1: If the user enters a wrong, empty-target, or duplicate URL, the system informs with an error |
| Search for documents | E2: If the user enters a non-keyword, the system informs with an error<br>E3: If the user enters a non-existent word, the system returns with an empty result |

# KEngine data requirements

- *a document has a title, some URLs, and a body; a body is a sequence of words*

→ Document: title, Url, body

- *many words are uninteresting and are not used as keywords*

  → Keyword, Non-keyword

  appears-in(Keyword,Document): frequency

◊ *a query begins by having a single keyword*

- *can be refined with another keyword*

→ Query: keywords

has(Query,Keyword)

- *a query result consists of matches, which are documents containing all the query keywords, ordered by the keyword frequencies*

→ Match: document, sum-freq

has(Query, Match)

refers-to(Match, Document)

# Non-functional requirements

- Constraints on the functions or emergent properties
- NFR may lead to other FRs
- Should be quantified when possible:
  - e.g. range of response time is 1-5 secs

# Types of NFR

- Performance
- Modifiability
- Reusability
- Delivery schedule

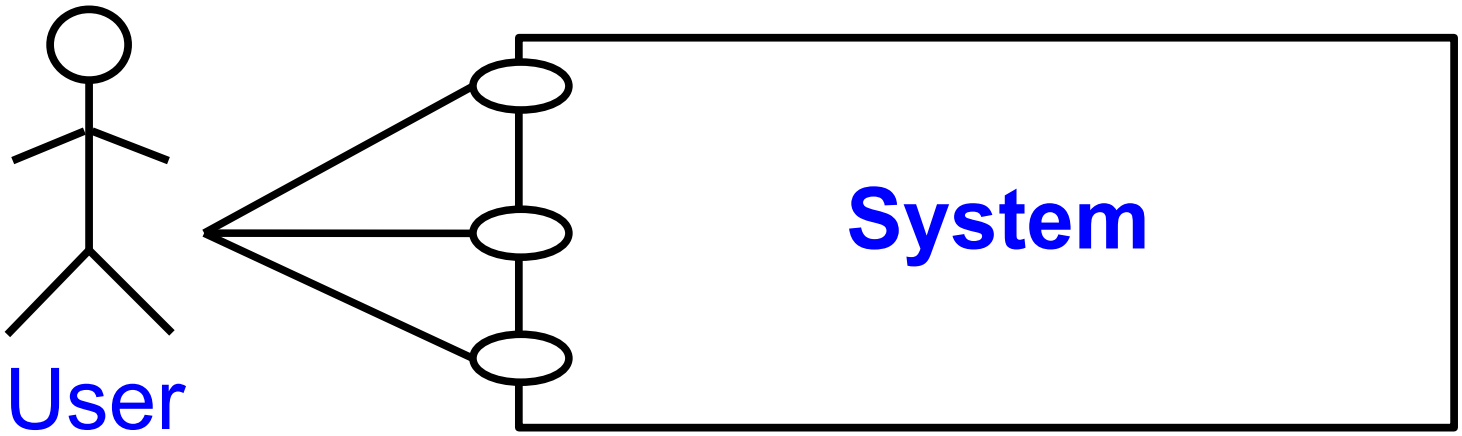
# Requirement capturing

- To capture the *detailed* requirements
- Techniques:
  - interview
  - document capturing
  - **use case**
  - prototyping

# Use case

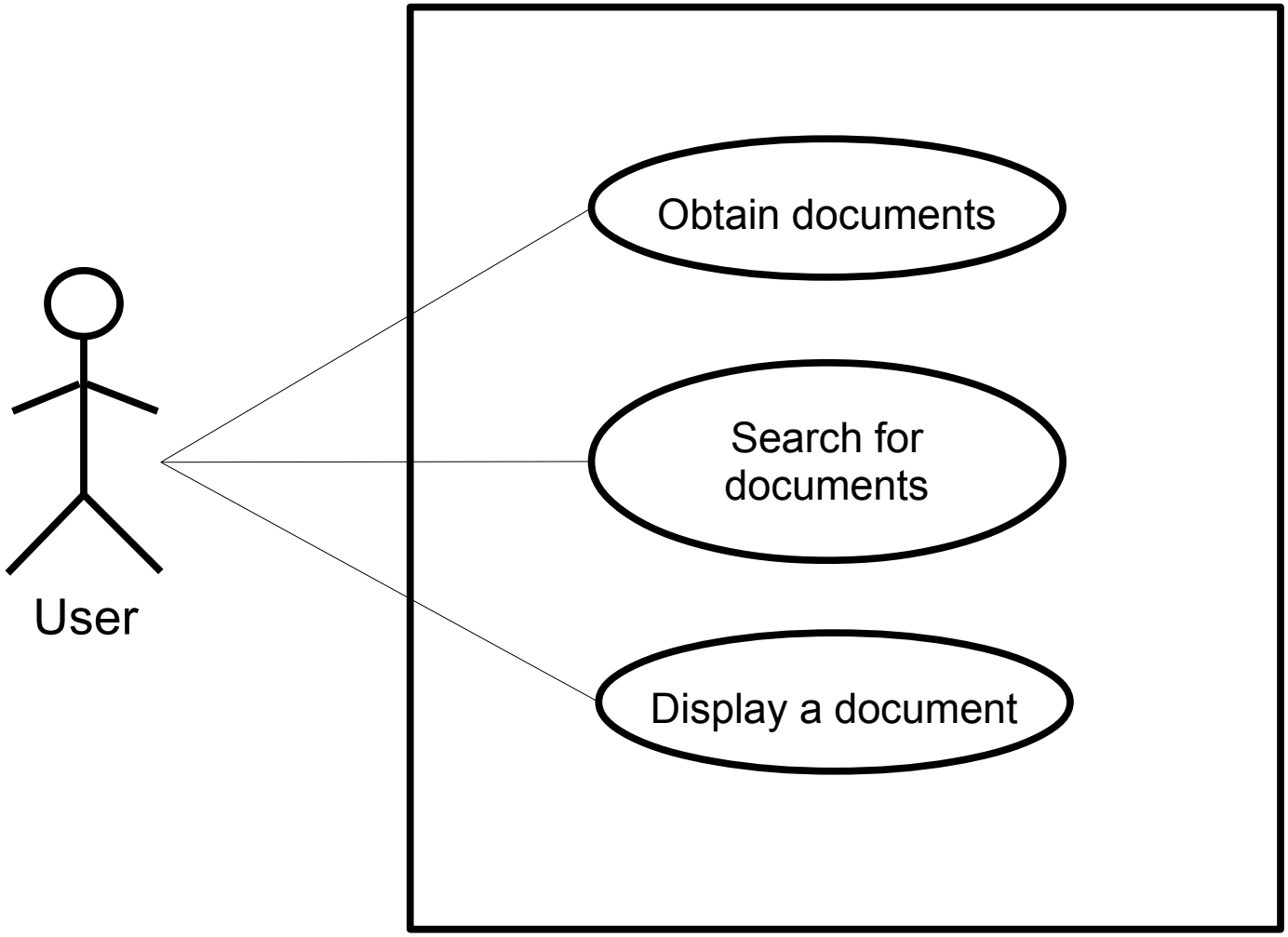
- One or more related user interactions with the system
  - an interaction is structured as a *scenario*
- Capture the details for each function
- Types: normal and extended
  - extended type include alternative and erroneous interactions
- Use cases can be linked
- Documented using a *use case description*

# UC illustration




User interacts with system by performing its functions

# Example: KEngine

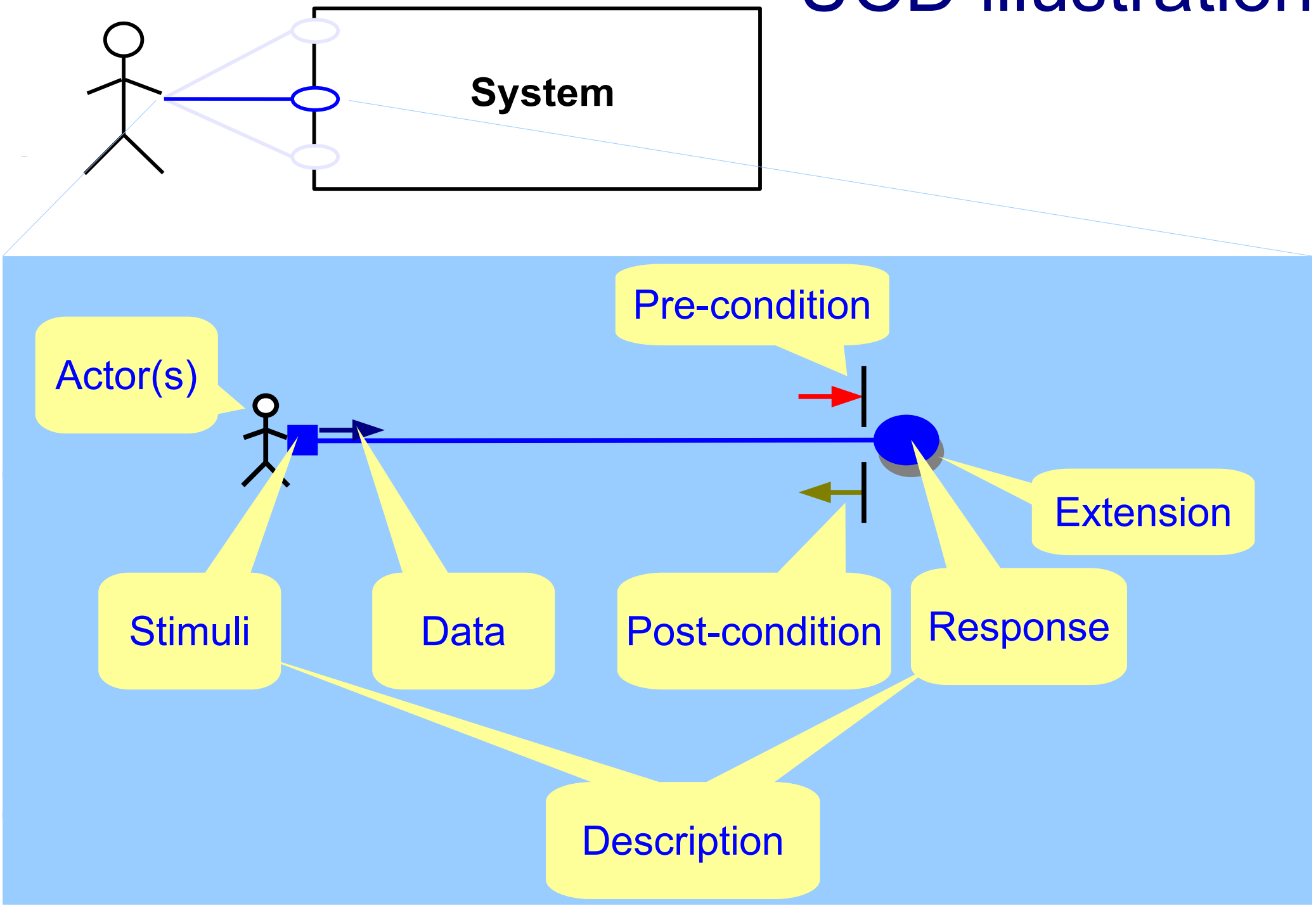



**KEngine System**

# Use case description (UCD)

| ***<Function name>*** | ***<Basic \| Extended>*** |
|---|---|
| **Actors** | *Name(s) of user(s) that interact* |
| **Description** | *Short description of the use case* |
| **Data** | *Data requirements* |
| **Stimuli** | *User action that causes the system to perform this function* |
| **Response** | *Description of function (what the system does in response to stimuli)* |
| **Pre-conditions** | *synonymous to REQUIRES* |
| **Post-conditions** | *synonymous to EFFECTS* |
| **Extension** | *errors (if any)* |

# UCD illustration



# Example: KEngine F2 (basic)

| **Search for documents** | **Basic** |
|---|---|
| **Actors** | User |
| **Description** | A user enters a keyword query and requests the system to execute it |
| **Data** | The input data include a keyword query |
| **Stimulus** | User command that requests the system to execute the query |

# (cont'd)

| | |
|---|---|
| **Response** | The system searches in the collection for the documents containing all the query keywords and return them as the result. Each document is considered a match, containing an aggregate of all the frequencies of the query keywords. |
| **Pre-conditions** | A document collection has been obtained and analysed to determine the keywords and non-keywords |
| **Post-conditions** | Query result containing the matching documents |

# F2 (extended)

| Search for documents | Extended |
|---|---|
| **Actors** | User |
| **Description** | A user enters a keyword query and requests the system to execute it |
| **Data** | The input data include a keyword query |
| **Stimulus** | User command that requests the system to execute the query |
| **Response** | The system searches in the collection for the documents containing all the query keywords and return them as the result. Each document is considered a match, containing an aggregate of all the frequencies of the query keywords. |
| **Pre-conditions** | A document collection has been obtained and analysed to determine the keywords and non-keywords |
| **Post-conditions** | Query result containing the matching documents |

# (cont'd)

| | |
|---|---|
| **Additional scenarios** | User enters a non-keyword → System informs with an error<br>User enters a non-existent word → System returns an empty result |

two erroneous interactions

# Summary

- Requirements may be functional or non-functional
- A requirement capturing technique is to use use case description (UCD)

# Questions?