Software Requirements Engineering

Zhiyuan Wan

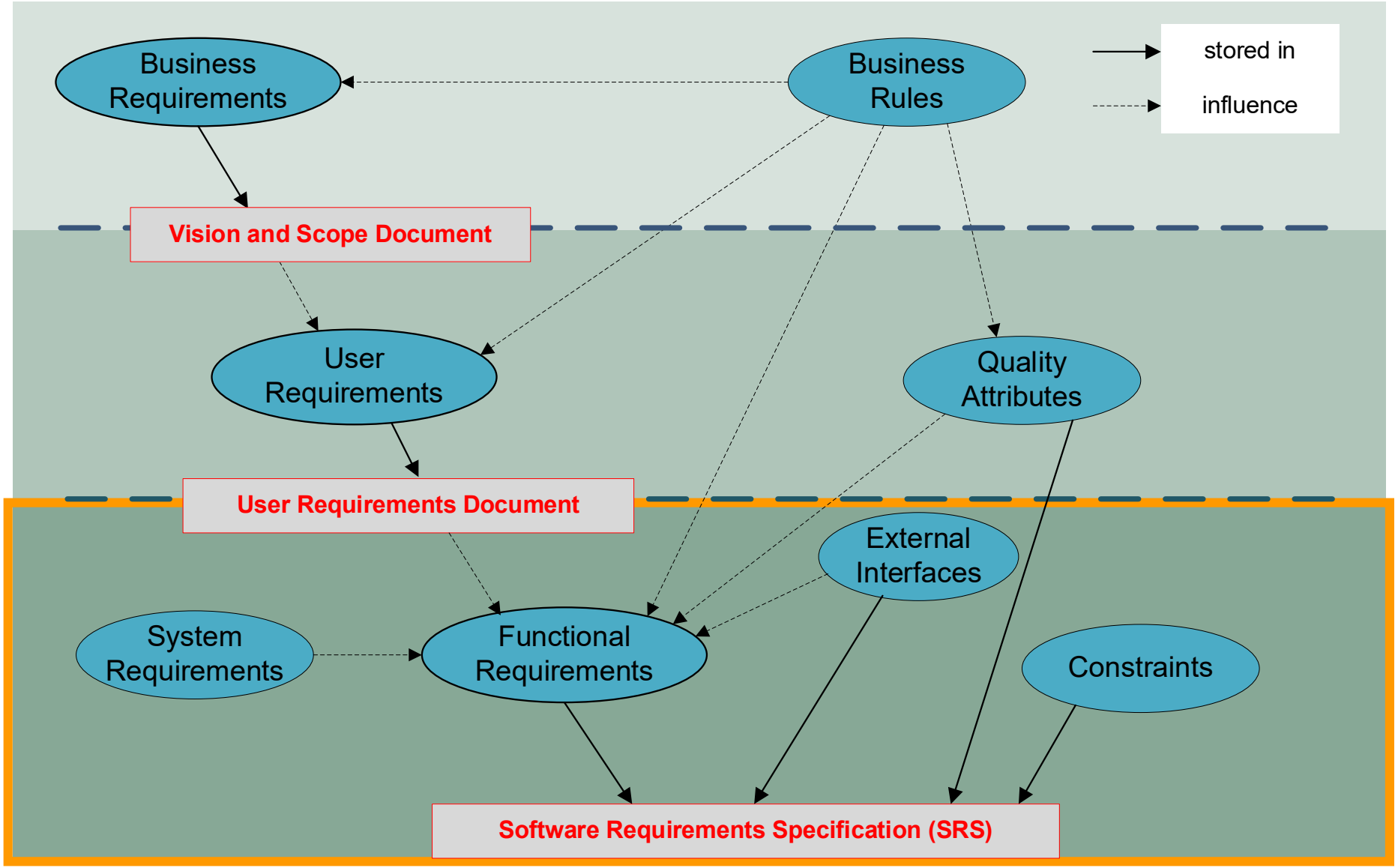
November 5, 2021

Agenda

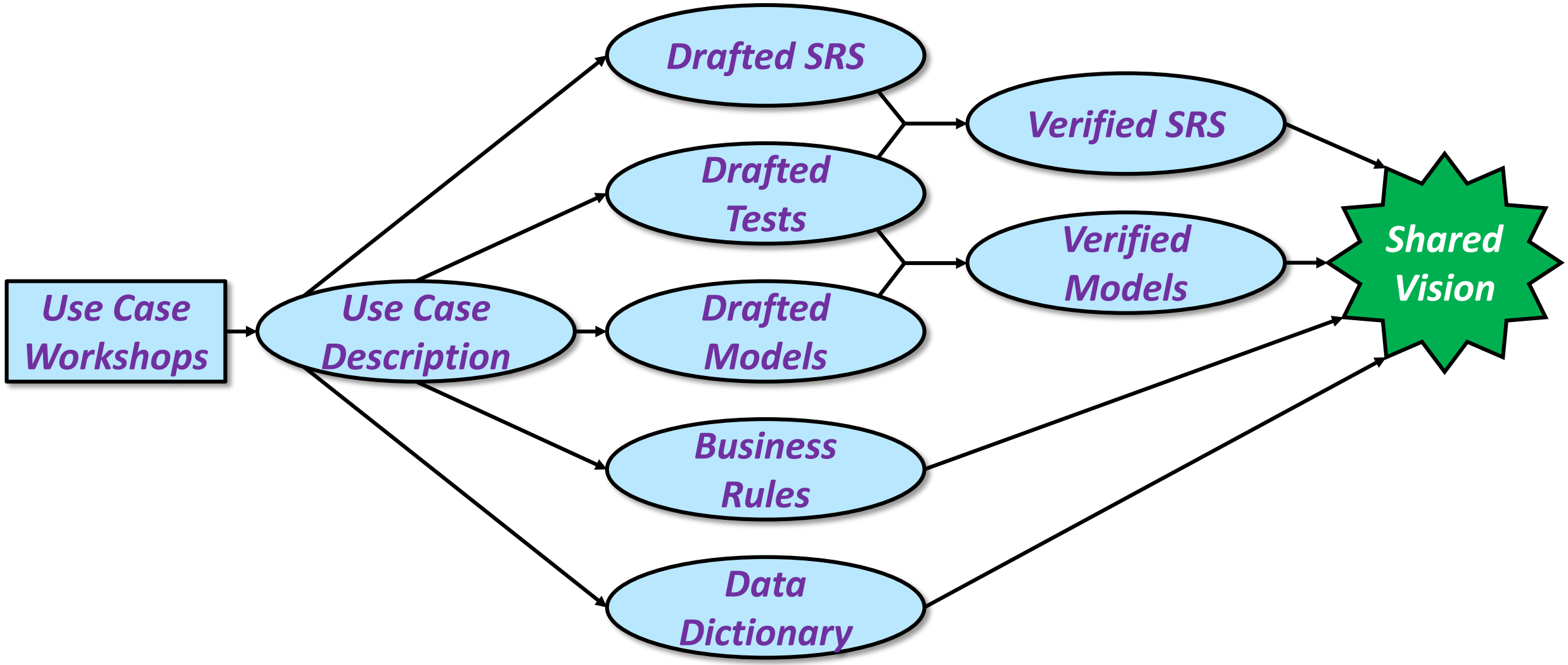
- Recap from last lesson
- Build the knowledge
- Project

Previously on software Requirement Engineering

Three levels of software requirements



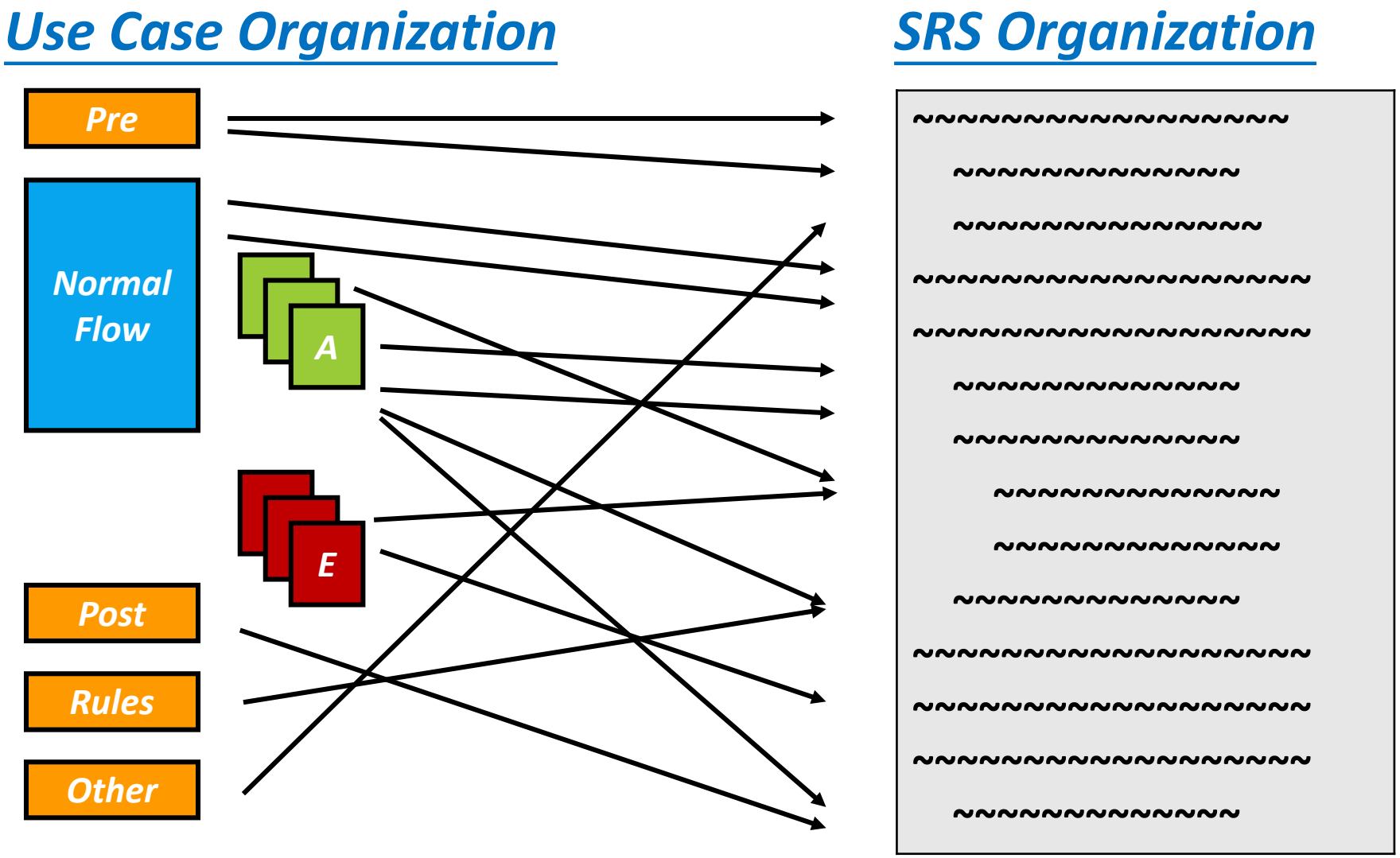
Products from Use Case Analysis



Use Case Template

ID and Name:			
Created By:		**Date Created:**	
Primary Actor:		**Secondary Actors:**	
Description:			
Trigger:			
Preconditions:			
Postconditions:			
Normal Flow:			
Alternative Flows:			
Exceptions:			
Priority:			
Frequency of Use:			
Business Rules:			
Other Information:			
Assumptions:			

Organizing Information: Use Case and SRS



Build the Knowledge

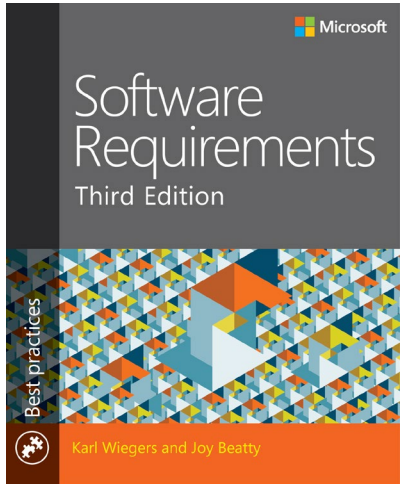
What we focus today

PART II: REQUIREMENTS DEVELOPMENT

CHAPTER 10 Documenting the requirements

CHAPTER 11 Writing excellent requirements

CHAPTER 12 A picture is worth 1024 words



Documenting the Requirements

What

Thought Process for Going From UCs -> FRs

- **Scan preconditions.**
  - ✓ what functionality is needed to test the precondition?
  - ✓ what happens if a precondition isn't satisfied?
- **Scan dialog steps.**
  - ✓ what functionality will let the dialog take place?
  - ✓ where are branches needed into alternative flows?
- **Scan exceptions.**
  - ✓ write exception-handling reqs with the associated functional req
  - ✓ what system recovery or reset operations might be needed?
- **Scan postconditions.**
  - ✓ how does the system satisfy visible and invisible postconditions?
- **Scan business rules.**
  - ✓ what functionality will enforce each business rule?

Software Requirements Specification Template
(adapted from IEEE Standard 830-1998)

1. Introduction
- 1.1 Purpose
- 1.2 Document Conventions
- 1.3 Intended Audience & Reading Suggestions
- 1.4 Project Scope
- 1.5 References

2. Overall Description
- 2.1 Product Perspective
- 2.2 Product Features
- 2.3 User Classes and Characteristics
- 2.4 Operating Environment
- 2.5 Design & Implementation Constraints
- 2.6 User Documentation
- 2.7 Assumptions and Dependencies

3. System Features
- 3.x System Feature X
- 3.x.1 Description and Priority
- 3.x.2 Stimulus/Response Sequences
- 3.x.3 Functional Requirements

4. External Interface Requirements
- 4.1 User Interfaces
- 4.2 Hardware Interfaces
- 4.3 Software Interfaces
- 4.4 Communications Interfaces

5. Other Non-functional Requirements
- 5.1 Performance Requirements
- 5.2 Safety Requirements
- 5.3 Security Requirements
- 5.4 Software Quality Attributes

6. Other Requirements

Appendix A: Glossary

Appendix B: Analysis Models

Appendix C: Issues List

IEEE Software Engineering standards: http://standards.ieee.org/software

Objectives for Functionality Specification

- Unambiguously describe what developers must implement
- Let testers know what system behaviors to expect
- Ensure that specified functionality will enable users to perform their necessary tasks
- Respect assumptions, constraints, and business rules
- Be able to trace functionality into design, code, and test elements

"The priest heard you finished the Functional Specification Document and wanted to witness the miracle."

https://www.modernanalyst.com/

Structure for Functional Requirements -1

- **From system's perspective:**

  Conditions: "When [some conditions are true] ..."

  Result: "the system shall" [do something]

  Qualifier: "... [response time or quality statement]."

- **Example:**

  "When the Patron indicates that he does not wish to order any more food items, the system shall display the food items ordered, the individual food item prices. and the payment amount within 1 second."

Structure for Functional Requirements - 2

- **From user's perspective:**

  User type: "The [user class name]…"

  Result type: "... shall be able to [verb]..."

  Object: "...[to something]..."

  Qualifier: "...[response time or quality statement]."

- **Example:**

  "The Patron shall be able to reorder any meal he had ordered within the previous six months, provided that all food Items in that order are available on the menu for the meal date."

How Much Detail Do You Need?

Less Detail ← → **More Detail**

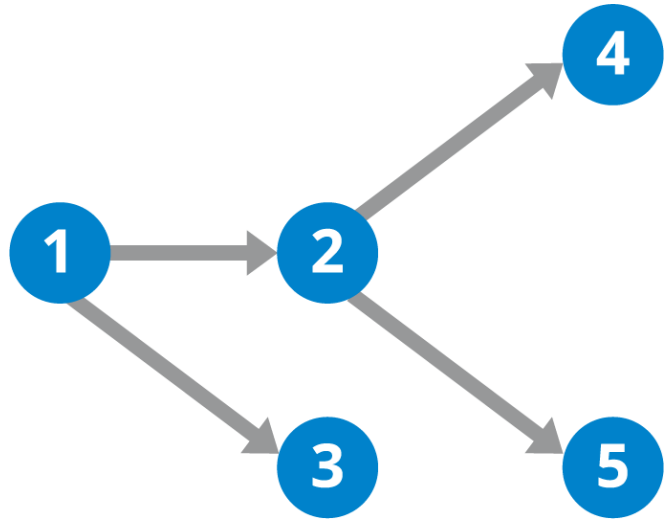
- ✓ Extensive customer involvement
- ✓ Developers have considerable domain experience
- ✓ Precedents are available
- ✓ Package solution will be used

Amount of detail needed in requirements

- ✓ Outsourced development
- ✓ Team is geographically dispersed
- ✓ Testing will be based on requirements
- ✓ Accurate estimates are needed
- ✓ Requirements traceability is needed

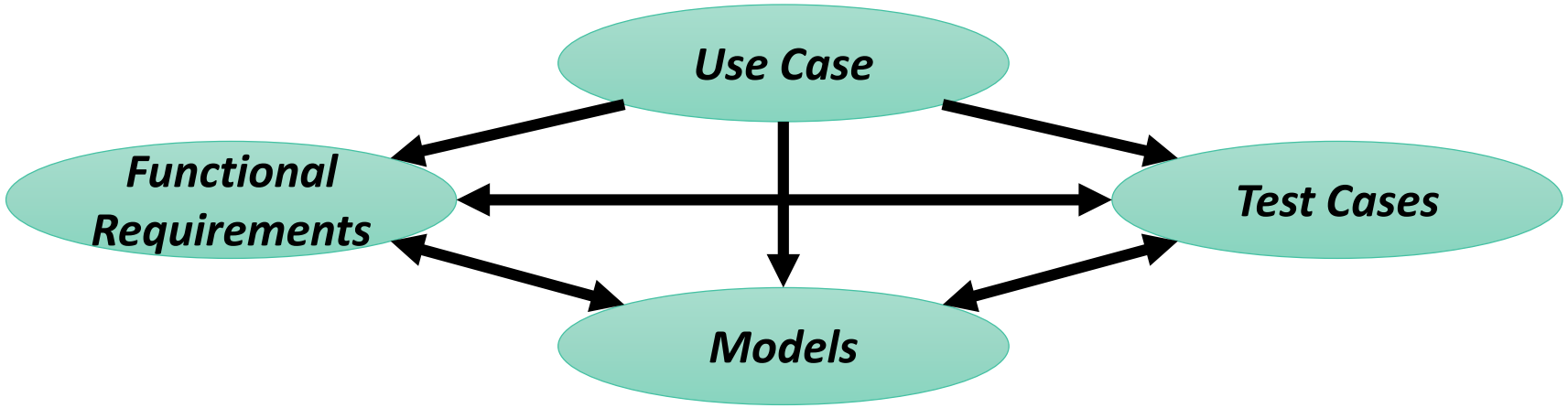
Using Multiple Requirements Views

- **Natural language text**
- Graphical analysis models
- Decision tables and decision trees
- Test cases
- Prototypes and screen designs
- Tables and structured lists
- Mathematical expressions
- Data dictionary
- Photographs, video clips, audio clips



Alternative Requirements Views

- A picture is worth 1024 words.
  - ✓ structured: context diagram, data flow diagram (DFD), entity-relationship diagram (ERD), state-transition diagram (STD)
  - ✓ object-oriented: class, sequence, interaction diagrams, etc.
- Models provide high-level view, SRS provides details.
- Conflicts between alternative views reveal errors.



A picture is worth 1024 words

How

From Voice of the Customer to Analysis Models

Type of word	Examples	Analysis model components
Noun	• People, organizations, software systems, data elements, or objects that exist	• External entities, data stores, or data flow (DFD) • Actors (use case diagram) • Entities or their attributes (ERD) • Lanes (swimlane diagram) • Objects with states (STD)
Verb	• Actions, things a user or system can do, or events that can take place	• Processes (DFD) • Process steps (swimlane diagram) • Use cases (***use case diagram***) • Relationships (ERD) • Transitions (STD) • Activities (activity diagram) • Events (event-response table)
Conditional	• Conditional logic statements, such as if/then	• Decisions (decision tree, decision table, or activity diagram) • Branching (swimlane diagram or activity diagram)

Select the Right Representations - 1

TABLE 12-2 Choosing the most appropriate representation techniques

Information depicted	Representation techniques
System external interfaces	■ *The context diagram* and *use case diagram* identify objects outside the system that connect to it. The context diagram and *data flow diagrams* illustrate the system inputs and outputs at a high level of abstraction. The *ecosystem map* identifies possible systems that interact, but includes some that do not interface directly as well. *Swimlane diagrams* show what happens in the interactions between systems. ■ External interface details can be recorded in input and output *file formats* or *report layouts*. Products that include both software and hardware components often have *interface specifications* with data attribute definitions, perhaps in the form of an application programming interface or specific input and output signals for a hardware device.
Business process flow	■ A top-level *data flow diagram* represents how a business process handles data at a high level of abstraction. *Swimlane diagrams* show the roles that participate in executing the various steps in a business process flow. ■ Refined levels of *data flow diagrams* or *swimlane diagrams* can represent business process flows in considerable detail. Similarly, *flowcharts* and *activity diagrams* can be used at either high or low levels of abstraction, although most commonly they are used to define the details of a process.
Data definitions and data object relationships	■ The *entity-relationship diagram* shows the logical relationships between data objects (entities). *Class diagrams* show the logical connections between object classes and the data associated with them. ■ The *data dictionary* contains detailed definitions of data structures and individual data items. Complex data objects are progressively broken down into their constituent data elements.

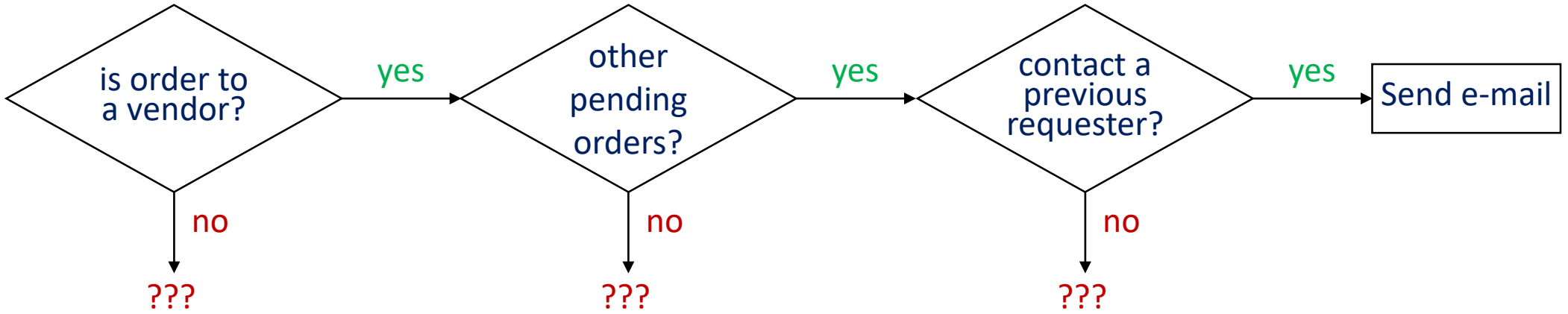
Select the right representations - 2

Information depicted	Representation techniques
System and object states	■ *State-transition diagrams* and *state tables* represent a high-abstraction view of the possible states of a system or object and the changes between states that can take place under certain circumstances. These models are helpful when multiple use cases can manipulate (and change the state of) certain objects. ■ Some analysts create an *event-response table* as a scoping tool, identifying external events that help define the product's scope boundary. You can also specify individual functional requirements with an event-response table by detailing how the system should behave in response to each combination of external event and system state. ■ *Functional requirements* provide the details that describe exactly what user and system behaviors lead to status changes.
Complex logic	■ A *decision tree* shows the possible outcomes from a set of related decisions or conditions. A *decision table* identifies the unique functional requirements associated with the various combinations of true and false outcomes for a series of decisions or conditions.
User interfaces	■ The *dialog map* provides a high-level view of a proposed or actual user interface, showing the various display elements and possible navigation pathways between them. ■ *Storyboards* and *low-fidelity prototypes* flesh out the dialog map by showing what each screen will contain without depicting precise details. *Display-action-response models* describe the display and behavior requirements of each screen. ■ *Detailed screen layouts* and *high-fidelity prototypes* show exactly how the display elements will look. *Data field definitions* and *user interface control descriptions* provide additional detail.

Example of Representation Techniques

Text: If an order is placed for a chemical to a vendor. the system shall check to see if there are any other pending orders for that chemical. If there are. the system shall display the vendor name. vendor catalog number. and the name of the person who placed each previous order. If the user wishes to contact any person who placed a previous order. the system shall allow the user to send that person an e-mail message.

Decision Tree:



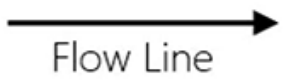
Data Flow Diagram (DFD) - 1

External Entity

An element that inputs data into an information system and / or retrieves data from the information system

Process

When an action takes place on data, potentially one of the information processes. A DFD contains multiple processes, each manipulating the data in their own way.



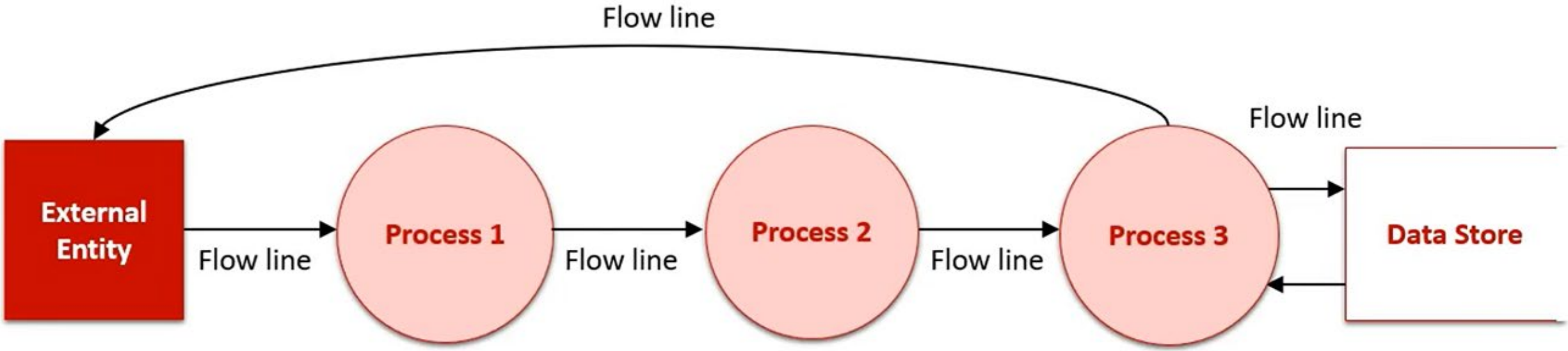
Flow Line

Illustrates the movement of data from one entity / process to another. A Data Flow line is supported by text stating what data is being sent / retrieved

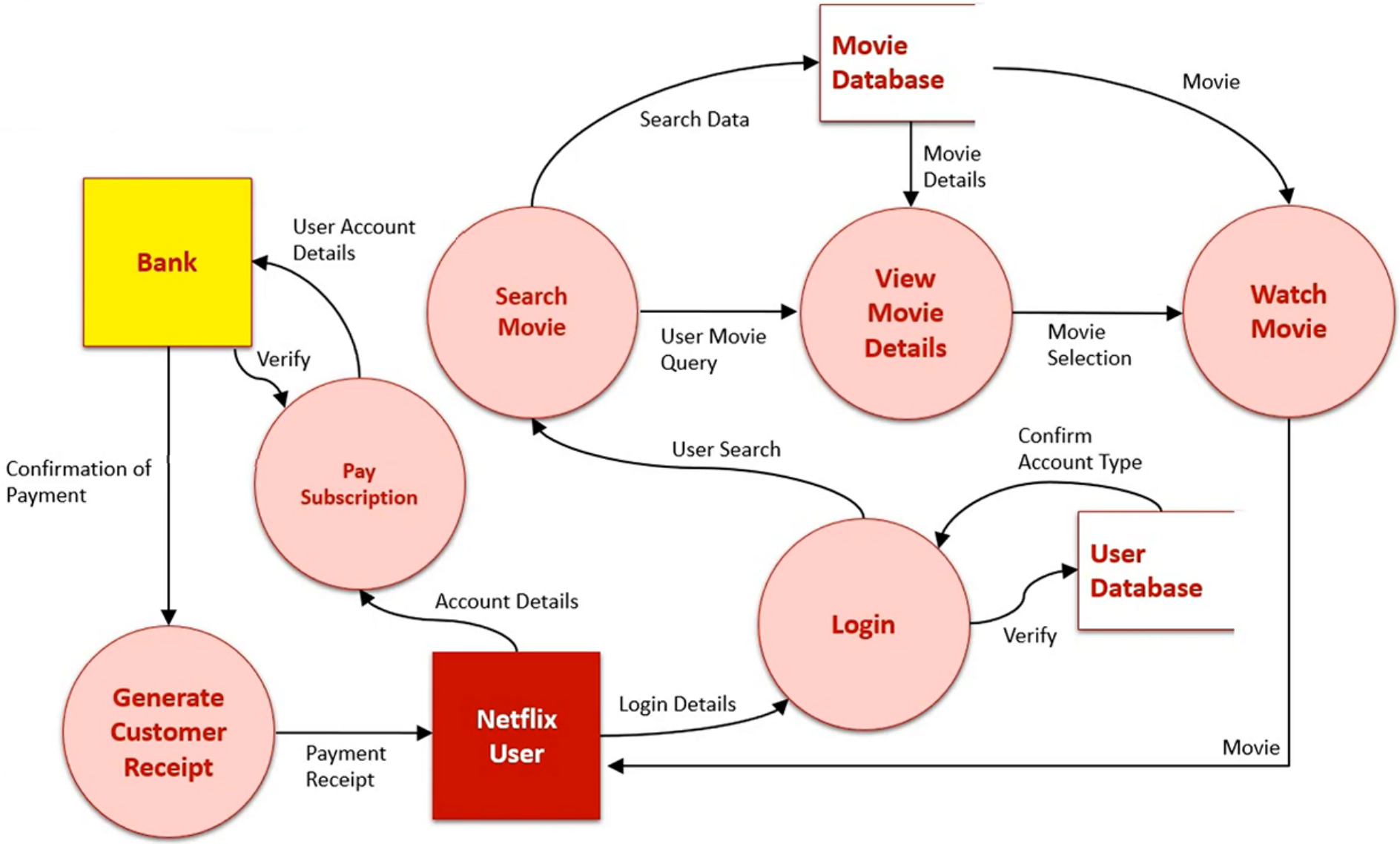
Data Store

A location where data is saved to or retrieved from, such as a database.

Data Flow Diagram (DFD) - 2



Netflix Data Flow Diagram



Writing excellent requirements

How

Characteristics of Excellent Requirements

- **Complete** nothing is missing; no "To Be Determined"s
- **Consistent** does not conflict with other requirements
- **Correct** accurately states a user or external need
- **Feasible** can be implemented within known constraints
- **Modifiable** can be easily changed, with history, when necessary
- **Necessary** documents something users really need
- **Prioritized** ranked as to importance of inclusion in product
- **Traceable** can be linked to system requirements, and to designs, code, and tests
- **Unambiguous** has only one possible meaning to all readers
- **Verifiable** correct implementation can be determined by testing, inspection, analysis, or demonstration

Tips for Writing Clear Requirements - 1

- ❑ Evaluate from the developer's perspective.
- ❑ Document in hierarchical, structured form.
- ❑ Keep sentences and paragraphs short and simple.
  - ✓avoid long narrative paragraphs
  - ✓use proper grammar, spelling, and punctuation
  - ✓use vocabulary of the business domain
- ❑ Document both expected behaviors and exception conditions.

https://www.modernanalyst.com/

Tips for Writing Clear Requirements - 2

- ❑ Avoid unnecessary design constraints.
- ❑ Write requirements at fine granularity.
  - ✓ decompose requirements until each is discretely testable
  - ✓ "and" and "or" suggest multiple requirements combined
- ❑ Be precise and specific, not vague and ambiguous.
  - ✓ use "**shall**" or "**must**," not "**should**," "**might** ", "**may**"
  - ✓ avoid ambiguous and subjective words

https://www.modernanalyst.com/

Some Weak Words to Avoid in Requirements

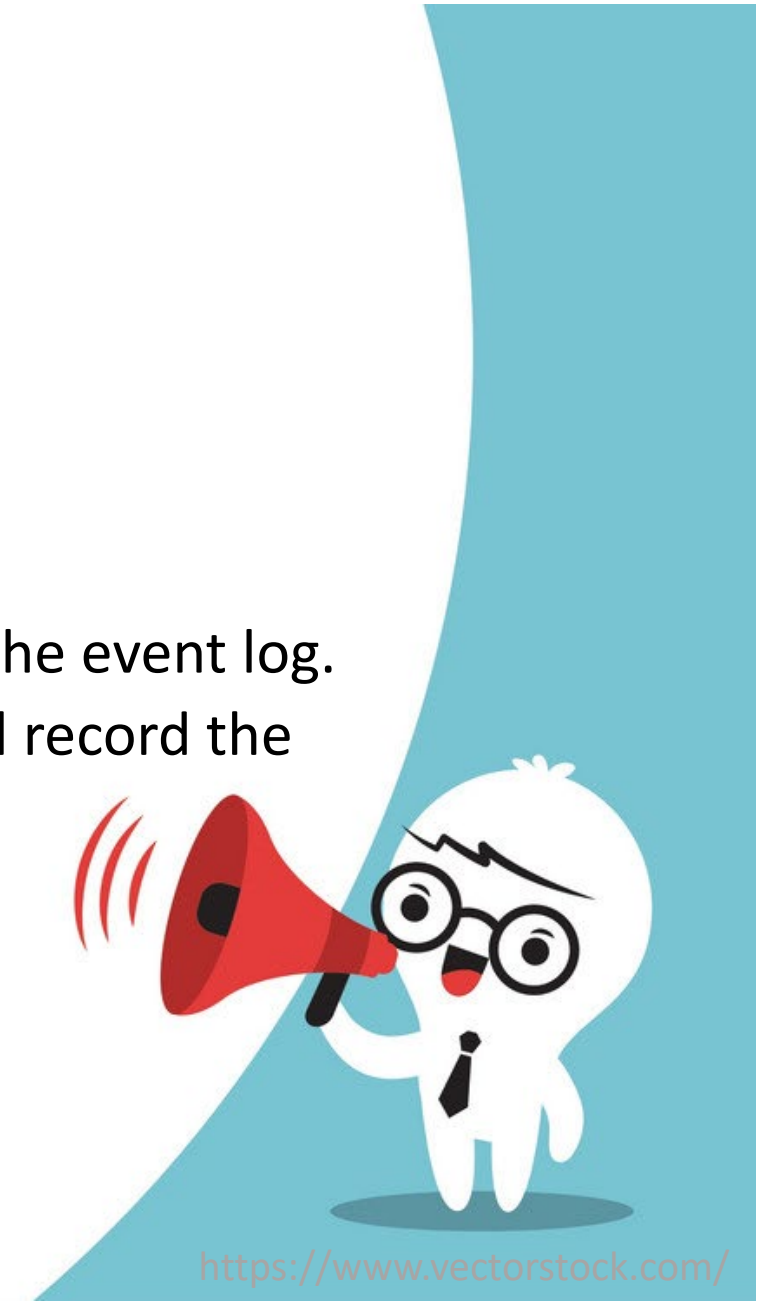
- minimize, maximize, optimize
- user-friendly, rapid, easy, simple, intuitive, efficient, flexible, robust
- seamless, transparent, graceful
- improved, state-of-the-art, superior
- sufficient, adequate, at least
- reasonable, where appropriate, to the extent possible, if necessary
- few, several, some, many
- etc., including, and/or
- optionally
- support

Writing in Active Voice

- **Use the active voice in requirements.**
  - ✓ shows which entity takes each action
  - ✓ communicates more directly than passive voice
- **Example:**

Passive: When the output state changes, it is logged in the event log.

Active: When the output state changes. the system shall record the new state in the event log.



Common Types of Requirements Ambiguity

- **Ambiguity of reference**
- **Scope of action**
- **Omissions**
  - ✓ causes without effects
  - ✓ effects without causes
  - ✓ complete omission
  - ✓ implicit cases
- **Ambiguous Logic**
  - ✓ nested ands, ors, nots
  - ✓ missing 'else’
  - ✓ implicit logical connectors
  - ✓ compound operators
- **Negation**
  - ✓ unnecessary negation
  - ✓ double or triple negation
- **Ambiguous Statements**
  - ✓ synonyms
  - ✓ ambiguous precedent
  - ✓ ambiguous timing
- **Poor Organization**
  - ✓ mixed causes and effects
  - ✓ random sequence
  - ✓ fragmented requirements
- **Boundary Ambiguity**

Examples of Avoiding Ambiguity - 1

❑ Negation

Before: All users with three or more accounts should not be migrated.

After: The system shall migrate only users having fewer than three accounts.

❑ Omissions

Before: The system shall display the user's defined bookmarks in a collapsible hierarchical tree structure

After:

Examples of Avoiding Ambiguity - 1

❑ Negation

Before: All users with three or more accounts should not be migrated.

After: The system shall migrate only users having fewer than three accounts.

❑ Omissions

Before: The system shall display the user's defined bookmarks in a collapsible hierarchical tree structure

After: The system shall display the users defined bookmarks in a collapsible and **expandable** hierarchical tree structure.

Examples of Avoiding Ambiguity - 2

- **Boundary Values**

Before:

1. If the amount of the cash refund is less than $50. the system shall open the cash register drawer.
2. If the amount of the cash refund is more than $50 and the user is not a supervisor. the system shall display a message: "Call a supervisor for this transaction. "

After:

Examples of Avoiding Ambiguity - 2

- **Boundary Values**

Before:

1. If the amount of the cash refund is less than $50. the system shall open the cash register drawer.
2. If the amount of the cash refund is more than $50 and the user is not a supervisor. the system shall display a message: "Call a supervisor for this transaction. "

After:

1. If the amount of the cash refund is less than ***or equal*** to $50, the system shall open the cash register drawer.
2. If the amount of the cash refund is more than S50 and the user is not a supervisor. the system shall display a message: "Call a supervisor for this transaction."

Examples of Avoiding Ambiguity - 3

- **Synonyms and Near Synonyms**
  - ✓ use terms consistently
  - ✓define terms in a glossary
- **Similar Sounding Words**
  - ✓ “Special Day carter tunes (default) will take priority over all configured individual caller settings that a customer has selected. However, If an individual has been assigned a Special Day caller tune for the same date, this will ***overwrite*** the Special Day caller tune”
- **Pronouns make the antecedents crystal clear**

Examples of Avoiding Ambiguity - 4

❑ Adverbs

- ✓ Provide a ***reasonably*** predictable end-user experience.
- ✓ Offer ***significantly*** better download times.
- ✓ Optimize upload and download to perform ***quickly***.
- ✓ Exposing information ***appropriately***...
- ✓ ***Generally*** incurs a 'per unit" cost...
- ✓ ...as ***expediently*** as possible...
- ✓ ***Occasionally*** (not very ***frequently***) there will be an error condition...
- ✓ others: easily, ideally, instantaneously, normally, optionally, periodically, rapidly, typically, usually

Project

Questions and Answers

Milestones in the Project (Updated)

ID	Milestone	Percentage	Deliverable (ZIP, PDF)	Due
1	Team Workflow	5%	**1. Document: (PDF)** <= 5 pages **2. Process files: (ZIP)** Snapshots of online meetings, meeting minutes, commit logs of the GitHub repo **3. Presentation: (PDF)** <= 15 minutes Transfer slides to PDF	24:00 Oct 6, 2021
2	Vision and Scope	10%		24:00 Oct 13, 2021
3	Software Requirements Specification and Mid-Term Presentation	20%		**24:00 Nov 17, 2021**
4	Design and Coding	35%		24:00 Dec 22, 2021
5	Final Report and Presentation	30%		24:00 Dec 29, 2021

Email to: zju_sre_21@163.com before due date;
Naming: PROJECT-MILESTONE[n]-[team_id]-[team_name]