

Requirement Engineering

Lecture 4: Requirements Documentation Part 2

Prof. Dr. Benjamin Leiding
Anant Sujatanagarjuna

General Requirements Engineering Process

Overview

Requirements Engineering					
Requirements Analysis				Requirements Management	
Elicitation	Negotiation	Documentation	Validation	Change Management	Tracing

Lecture 4: Requirements Documentation

Content

1. Types of Requirements
2. Textual Requirements Specification



TYPES OF REQUIREMENTS

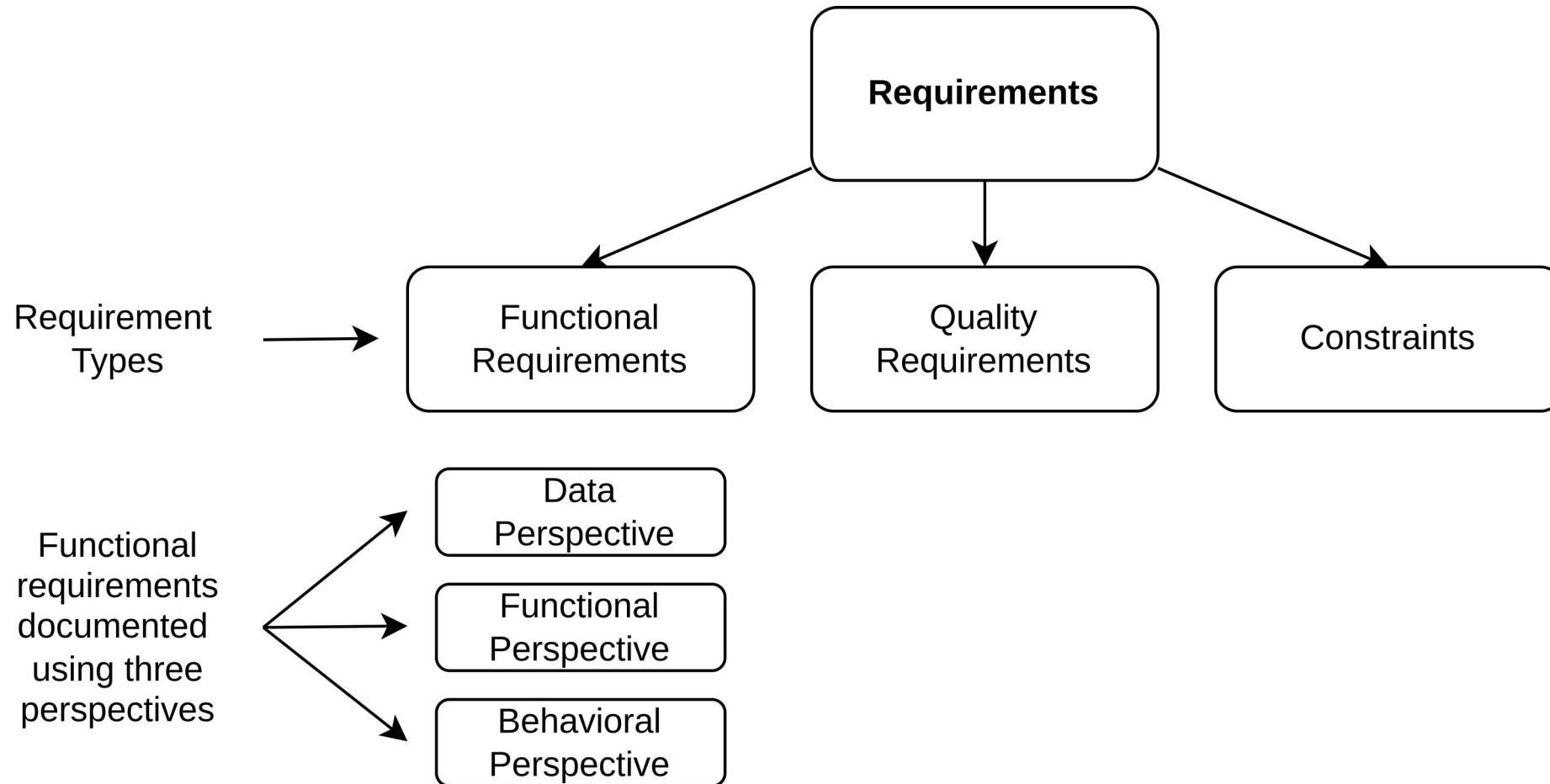
Types of Requirements

Motivation

- Different types of requirements must be documented for a complete requirements documentation.
- These requirements types differ with respect to:
 - adequate specification techniques
 - their importance for different types of systems

Types of Requirements

Functional Requirements



Types of Requirements / Functional Requirements

Data Perspective

- All systems need to deal with data
 - Data on customers, articles, etc.
 - Multimedia, e.g., videos, songs, etc.
 - ...
- Information must be adequately structured and represented:
 - Which information / data items are relevant to the system?
 - Which information / data items are at the boundary of the system?

Types of Requirements / Functional Requirements

Data Perspective - Representing Data

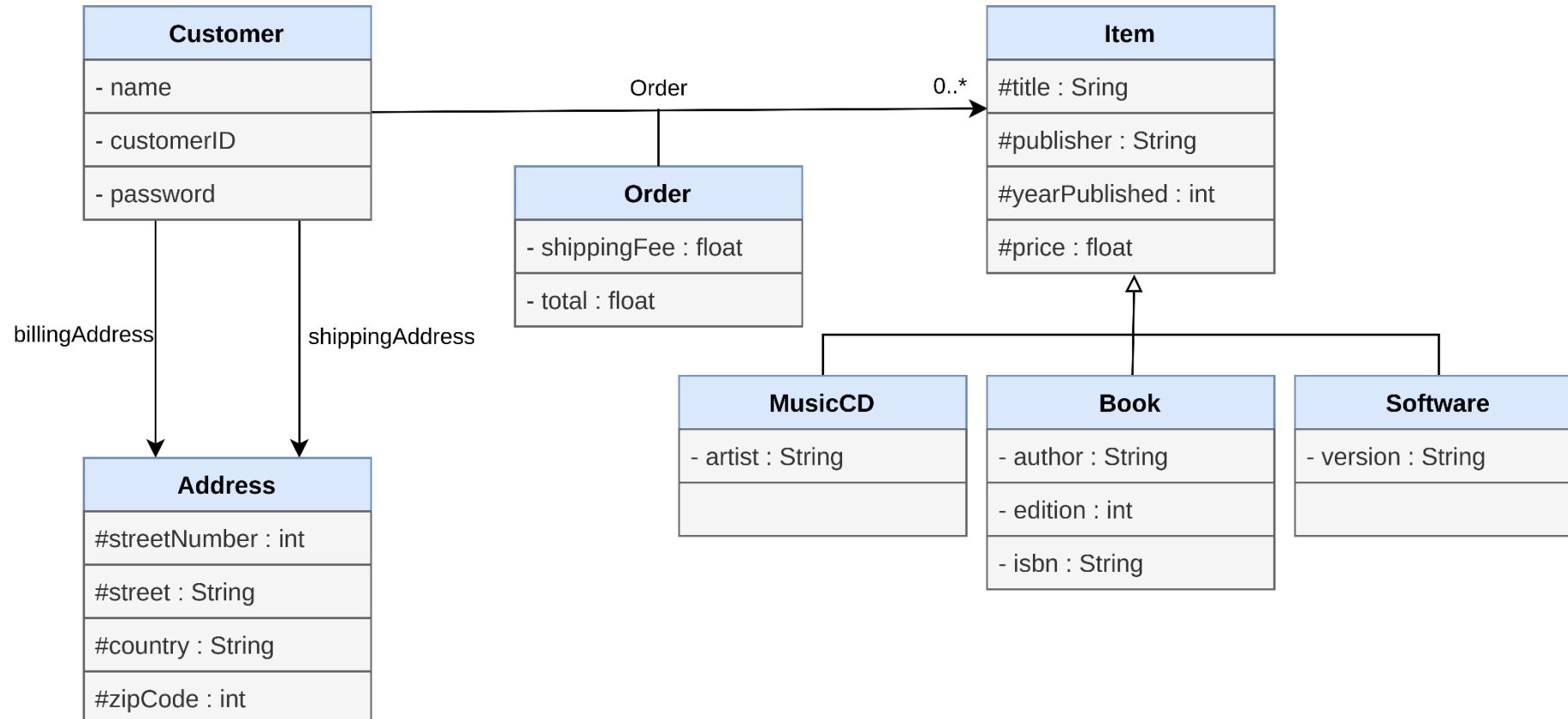
- UML class diagrams
- Context diagrams
- Data dictionary

Note:

- the data specified in the requirements need not be directly related to the implementation
 - same information, but different structure possible
 - e.g. attributes versus classes may change strongly
- In information systems understanding the data is a key driver !

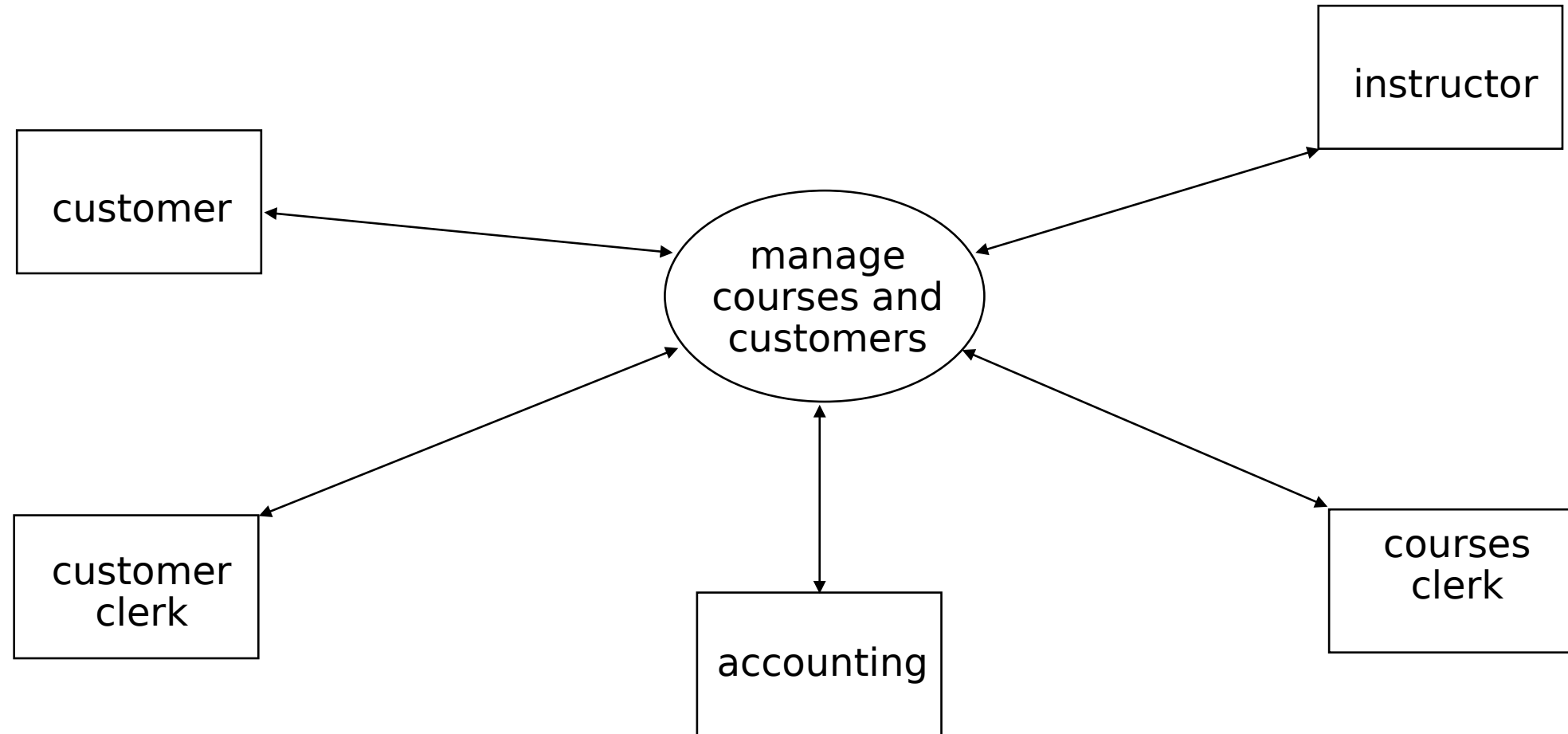
Types of Requirements / Functional Requirements

Data Perspective - Book Shop UML Example



Types of Requirements / Functional Requirements

Data Perspective - Context Diagram Example



Types of Requirements / Functional Requirements

Behavioral Perspective

Behavioral requirements describe what a system will do (with the data):

- how input information is transformed into state information and output information
- sequences of interaction of the software system with its environment (people, software, hardware)

System behavior is important on various levels:

- **Business processes** - describe the fundamental flow of activities in an enterprise
- **Task level** - describe the interaction of people with a software system on a coarse grained level (e.g., define new customer)
- **Stimulus / response** - describe interactions

Types of Requirements / Functional Requirements

Behavioral Perspective – Specification Techniques

Many different techniques were developed for specifying this:

- Textual Use Cases
- Business Process Modeling Languages
- Scenario-Based Modeling Approaches
- Event-Based Modeling Techniques

The techniques can be categorized along the following dimensions:

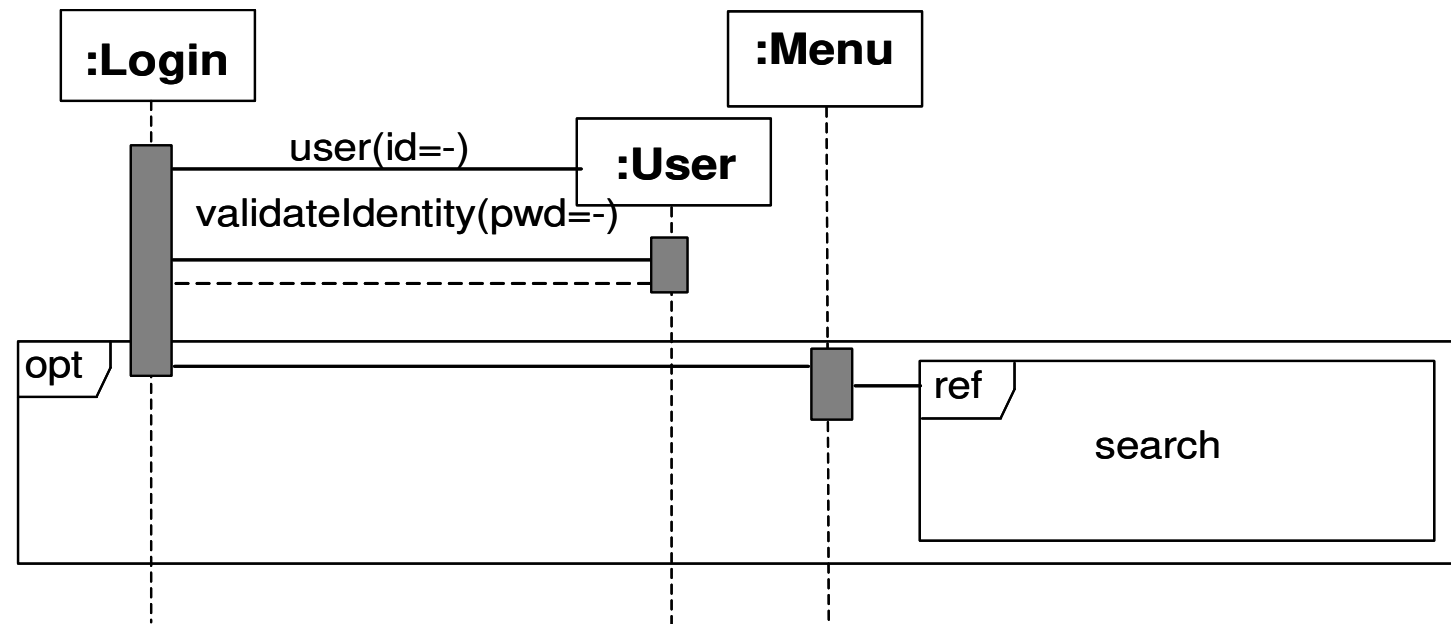
- **data-flow** (-transformation) vs. **stimulus-/response**
- **complete** description vs. **prototypical** description

Types of Requirements / Functional Requirements

Behavioral Perspective - Specification Techniques

The UML provides different approaches:

- Use Case Diagram
- State Machine Diagram
- Activity Diagram
- Sequence Diagram



Types of Requirements

Non-Functional & Quality Requirements - Definition

- 1 Quality requirements **define qualitative attributes** of the whole system, a single function or a group of functions, i.e. how good a system shall do the things it is supposed to do.
- 2 Non-functional requirements are used to **encompass all kinds of *not functional requirements*** for a system:
 - quality requirements should be related to the functional requirement or group of requirements they are relevant to
 - development constraints should be captured separately
 - project aspects should be clearly separated from product aspects

Types of Requirements

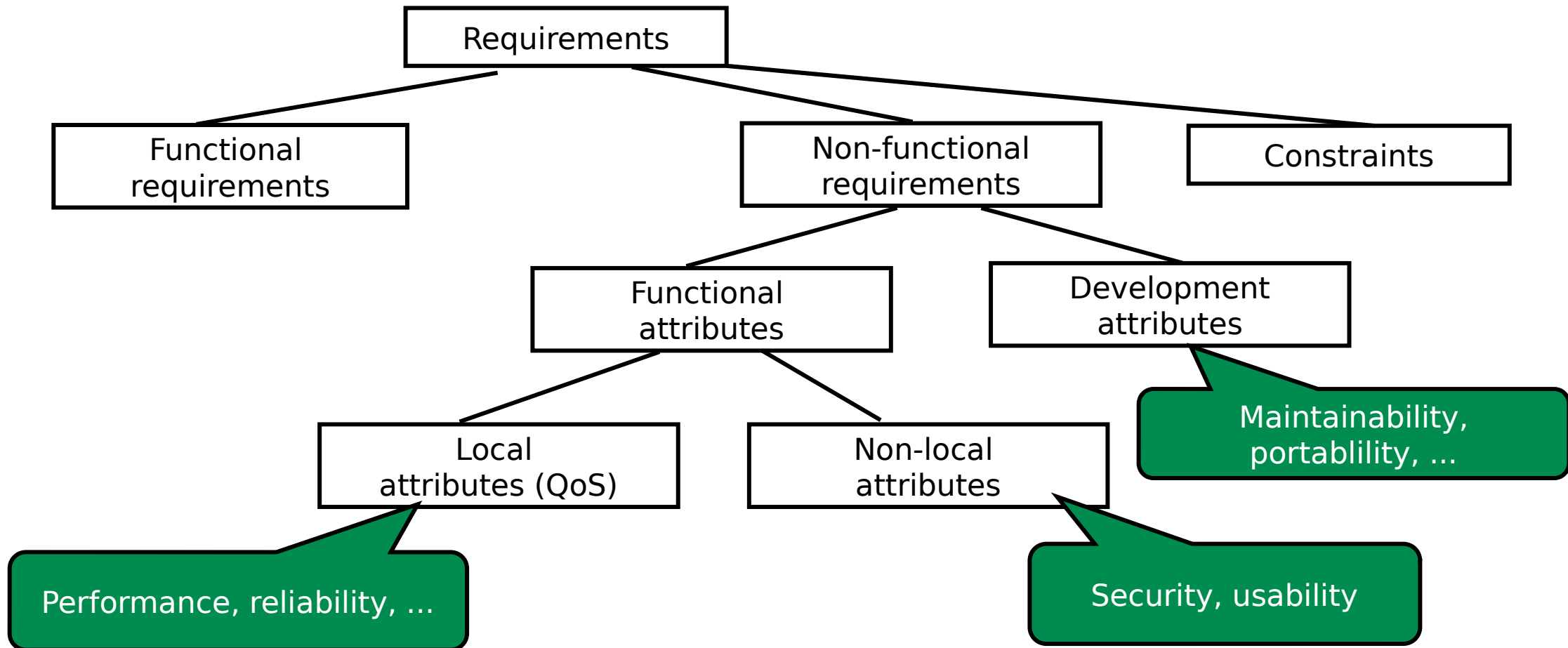
Non-Functional & Quality Requirements - Definition

- 1 Quality requirements **define qualitative attributes** of the whole system, a single function or a group of functions, i.e. how good a system shall do the things it is supposed to do.
- 2 Non-functional requirements are used to **encompass all kinds of *not functional requirements*** for a system:
 - quality requirements should be related to the functional requirement or group of requirements they are relevant to
 - development constraints should be captured separately
 - project aspects should be clearly separated from product aspects

The term non-functional requirements is depreciated (according to IEEE)

Types of Requirements

Non-Functional & Quality Requirements - Kinds of Requirements



Types of Requirements

Non-Functional & Quality Requirements – Product Quality (ISO 9126 / DIN 66272)

- Functionality
 - Adequacy
 - Security
 - Precision of calculation
 - Interoperability
 - Conformity with standards
- Reliability
 - Maturation
 - Fault tolerance
 - Recovery
- Usability
 - Comprehensibility
 - Learnability
 - Operability

Types of Requirements

Non-Functional & Quality Requirements – Product Quality (ISO 9126 / DIN 66272)

- Efficiency
 - Time response
 - Resource Consumption
- Changeability
 - Analyzability
 - Modifiability
 - Stability
 - Verifiability
- Portability
 - Adaptivity
 - Installability
 - Conformity with standards
 - Replaceability

Types of Requirements

Non-Functional & Quality Requirements - Example „Performance“

- User level → The user can create accounts with only **two** interactions
- System task level → The creation of an account (pressing of the „system availability“ button) takes max. 0.5 seconds

→ **Derived non-functional requirement result from the interplay between both levels.**

Types of Requirements

Interface Requirements

- The interface takes apart interior and exterior
- Interfaces are defined by the project context
- Different types of interfaces:
 - User interfaces (Human-Machine Interface)
 - Software Interfaces
 - Hardware-related interfaces

Types of Requirements

Interface Requirements - User Interface

- Interface Description must describe
 - Layout
 - Look & Feel
 - Category of Interface (WIMP, ASCII-based, tactile, ...)
 - Interaction sequences
 - ..

- Usability aspects are specific to this type of interface
 - Person (e.g., impairments, knowledge)
 - Situation
 - Task
 - ...

Types of Requirements

Interface Requirements – Software Interface

- Interfaces to other software interfaces are defined based on
 - Identification of service, i.e., how to find it
 - The protocol (how to interact)
 - The data format(s), e.g., how to exchange data

- Typically use of standard protocols, like
 - Web-Service
 - HTTP
 - ...

But also possible → Data file is written to a specific location and read by another program

Types of Requirements

Interface Requirements – Hardware Interface

Hardware interfaces are often:

- Time critical
 - Protocol specification must include timing information

- Specified close to hardware (e.g. addressing)
 - Hardware-based → service identification may be given in bits and bytes

- Other than that, usually hardware interfaces are like a software interface
- Mapping software information to the physical world is done by hardware!



TEXTUAL REQUIREMENTS SPECIFICATION

Lecture 4: Requirements Documentation

Content

1. Types of Requirements
2. Textual Requirements Specification
 - 1. Ambiguity**
 2. Guidelines
 3. Syntactic Requirements Patterns

Textual Requirements Specification

Advantages of Natural Language

Three essential advantages

- Universal
 - Can be used in any problem area or domain
- Flexible
 - Allows arbitrary abstractions and refinements
- Comprehensible
 - Can (potentially) be understood by any stakeholder

Textual Requirements Specification

Mixing Concepts

- Mixing of the **three** perspectives (data/structural, function, behavioral) in functional requirements
 - Often even mixed with quality requirements
 - Example
 - The glass break detector at the window detects that the pane has been damaged, the system shall inform the security service within 2 seconds at the least.
 - Structural: glass break detector, window, pane, system, security service
 - Function: detects, inform the security service
 - Behavior: if damaged, shall inform
 - Quality: 2 seconds
- **Mixing concepts is a bad idea**

Textual Requirements Specification

Separation of Functional and Quality Aspects

At least separate functional and quality aspects

- Functional
 - The glass break detector at the window shall detect if the glass pane is damaged.
 - If the detector detects damage to the pane, the system shall inform the security service.
- Quality
 - The system shall inform the security service within 2 seconds after detecting the damage.

Textual Requirements Specification

Ambiguity

A requirement is **ambiguous**, if it allows more than one interpretation even though the relevant context (other requirements, application domain, software system) is known.

Textual Requirements Specification

Ambiguity - Why should we care?

- Ambiguity is a common problem
- Ambiguity is often overlooked, as an interpretation is chosen unconsciously
 - Cause: Ambiguity as „under-specification“ is a typical phenomenon of natural language. The solution of ambiguity is an (often unconscious) cognitive process taking context (e.g. shared situation) or other cues (e.g. nonverbal) into account.
 - The „most likely“ interpretation of a requirement is chosen unconsciously, thus the interpretation causing the least contradictions with already known requirements, domain attributes or standards is chosen.
 - Because requirements can be controversial, this – in contrast to the common, verbal everyday communication – is not an optimal strategy! Contradictions must be discussed with the parties and must be solved.
- Ambiguity can be a sign for incompleteness!

Textual Requirements Specification

Ambiguity - Impact on Software Engineering

- Consequences show up very late
 - During integration of software components
 - During acceptance test
 - During usage of the software
- Are ambiguous requirements a frequent problem?
- Result of a survey with specification techniques:
 - Omissions and conflicts in specifications are noticed more often than ambiguities
 - Ambiguities are rather self-interpreted and more often misinterpreted than other types of defects
 - RE specific ambiguity: a frequent problem
 - Linguistic ambiguity: a rare problem

Textual Requirements Specification

Ambiguity - Categories

Conscious ambiguity:

- Client wants to keep requirements open e.g. **usual** in public projects

Unconscious ambiguity:

- Client expects a certain interpretation of the requirement, ambiguity occurs as the expectations of customer and client are not shared

Linguistic ambiguity:

- Inherent attributes of the natural language „Flying airplanes can be dangerous”

RE specific ambiguity:

- Arises from interpretation of a requirement via background knowledge (other requirements, domain, etc.)

Textual Requirements Specification

Ambiguity - Types of Ambiguities

Vagueness:

- Continuum of interpretations, diffuse classification, summarized version of the interpretation available
 - The text editor has to respond to user input in the adequate time
 - *Are 10 seconds still adequate?*

Generality:

- Continuum of interpretations, but exact classification, summarized version of the interpretation is available
 - The ATM system shall increase the market coverage of the bank company XYZ by at least 5%
 - *No charge for ATM transactions, user interface should require as few user interactions as possible ...*

Textual Requirements Specification

Ambiguity – Types of Ambiguities

Genuine Ambiguity:

- Countable number of interpretations, no summarized version of the interpretation available, thus immediate clarification needed
 - Lexical: A term with several, in most cases related meanings
 - When the user presses the L- and R-button simultaneously, alarm is turned off → *The current alarm or the ability to sound alarms?*
 - Syntactic: Structure of a sentence is not clear without ambiguity
 - The customer enters a card with a code → *Is the code read from the card or is it typed in?*

Textual Requirements Specification

Ambiguity – Types of Ambiguities

- Semantic: A sentence can be translated into several logic terms
 - An alarm must be triggered if an aircraft is identified as hostile and has an unknown mission or in case the aircraft is able to reach the protected airspace within 5 minutes → *Is the „and“ or the „or“ the stronger binding operator?*
- Referential: A reference to an object is ambiguous to a previous sentence or subordinate clause. Is caused by nouns and pronouns.
 - The customer enters a card and a numeric personal code. If it is not valid then the ATM rejects the card. → *Card or code not valid?*
 - - [...] The product shall show all roads predicted to freeze. Reference of “all roads”?*

Textual Requirements Specification

Ambiguity - Types of Ambiguities

- Discourse ambiguity = A requirement is ambiguous in relation to other requirements.
- Example 1:
 - (A1) *When the XYZ button is pressed, the Head-up Display (HUD) shows the aircraft's current coordinates.*
 - (A2) *When the aircraft is not airborne, the HUD shows the current weather conditions.*
 - *Will the coordinates be displayed if the XYZ button is pressed and the aircraft is currently not airborne?*
- Example 2:
 - *The first dunning letter has to be created after 2 weeks and the second after 4 weeks. At that time the system is also sending a notice to the responsible official in charge. → Is the notice send after 2 or after 4 weeks? (or after 6 weeks?)*

Textual Requirements Specification

Typical Quality Problems

- Most requirements documentation is still done using text
- Typical quality problems of requirements
 - **Too restrictive:** requirements are described that unnecessarily restrict the range of possible interpretations
 - **Unnecessary:** single users request highly specialized functions, or the requirement does not contribute to the software systems goals.
 - **Inconsistent:** with goals of the software system, standards, directives, etc.
 - **Redundant:** with other information (in the requirements document)

Textual Requirements Specification

Typical Quality Problems

- Most requirements documentation is still done using text
- Typical quality problems of requirements
 - **Too restrictive:** requirements are described that unnecessarily restrict the range of possible interpretations
 - **Unnecessary:** single users request highly specialized functions, or the requirement does not contribute to the software systems goals.
 - **Inconsistent:** with goals of the software system, standards, directives, etc.
 - **Redundant:** with other information (in the requirements document)

→ *Style Guide*

Lecture 4: Requirements Documentation

Content

1. Types of Requirements
2. Textual Requirements Specification
 1. Ambiguity
 - 2. Guidelines**
 3. Syntactic Requirements Patterns

Textual Requirements Specification

Guidelines - Style Guide for the Specification of Requirements

Objectives:

- Requirements are easier to read and thus easier to understand
- Our style guide handles the most frequent problems, project-specific extensions may be reasonable
- Directives should be consolidated in a company-specific style guide

Textual Requirements Specification

Guidelines - Style Guide

- Short sentences, because of the limitation of the human short-term memory
- Describe only one requirement per sentence, avoid „and“
- Avoid jargon, use abbreviations sparingly
- Short paragraphs (max. 7 sentences)
- Use lists, instead of listing sentences
- Use terminology consistent; repetition of words is welcome!
- Avoid nested logic terms
 - If X or Y is given in case Z, but not..
 - ⇒ Use pseudo code or decision tables

Textual Requirements Specification

Guidelines - Example

Bad

Users attempting to access the ABC database should be reminded by a system message that will be acknowledged and by page headings on all reports that the data is sensitive, and access is limited by their system privileges.

Good

- 4.1 The system shall notify users attempting to access the ABC database that
- The ABC data is classified “sensitive”
 - Access to the ABC data is limited according to the user’s system privileges
 - Page headings on all reports generated using the ABC database must state that the report contains sensitive information
- 4.1.1 The system shall require the user to acknowledge the notification before being allowed to access the ABC database.

Textual Requirements Specification

Guidelines - Style Guide

- Use words like 'must', 'can', 'ought', 'should', 'is', etc. carefully
 - *Either*: precise definition: 'must', 'ought' show that the requirement is mandatory, etc.
 - *Or*: separate mandatory from optional requirements through a definition of a respective attribute or through a chapter heading
- Use active instead of passive
 - *Wrong*: a result is displayed
 - *Right*: the system displays the result (thus the actor is obvious!)
- Illustrate complex dependencies with graphics
- Use precise references
- Use automatic spellchecker

Textual Requirements Specification

Guidelines - Style Guide

- Express requirements so they are testable. Thus it is possible to check whether or not the system meets the requirements
 - Is it possible to create a test case for requirement X?
- State rationale for each requirement
 - The rationale is important as a basis for deciding upon changes or omissions of requirements during development
- Explanations in requirements are confusing
 - Negative example: “To enable an experienced user to work efficiently, the access authorization is also checked on double-clicking a list item and if this authorization is valid, the customer-specific data will be displayed in ‘Access’ field. In case the SQL-query returns an error code (-1), ...”
 - Better solution: Make explanations explicit

Textual Requirements Specification

Guidelines - Style Guide

- Avoid generalities
 - Leads to ambiguities → Example Tamagotchi: “On clicking the R-button the selected function is canceled.” Is this also true for the time function?
 - Seems boring if it has platitude characteristics → Example: “Input masks should be displayed entirely on screen. Scrolling should be avoided if possible. That is a principle of graphical user-interface design!”
- Document the sources (persons) of all requirements
 - For a large number of requirements or after a certain period of time, it is difficult to remember a source, if a requirement must be changed.

Textual Requirements Specification

Guidelines - Technical Terms

- Why should technical terms be defined?
- The advantage is to avoid misunderstandings caused by the following phenomena:
 - Unclear terms. Meaning is unclear to the requirement engineer (e.g., “butterfly valve”)
 - Ordinary terms may have special meanings to clients/users (“article”, “call”)
 - Different terms for the same „thing“ (synonyms) used by different sources or because the vocabulary of concepts of the client is not yet defined
 - Same term for related, but still different „things“ (polysemy) e.g. “school” = the institution *or* specific school (e.g., Werner v. Siemens Schule in Hildesheim)

Textual Requirements Specification

Guidelines - Technical Terms

Choose terms appropriate for the readers

Example → ISDN phone

- *For the hardware engineer:* key codes and activation of the LCD display
- *For the interface designer:* key sequences and masks on the LCD display
- *For the user of the telephone:* functions like call forwarding
- The correct description level is the one, that suits the expectations of the requirements-document reader

Lecture 4: Requirements Documentation

Content

1. Types of Requirements
2. Textual Requirements Specification
 1. Ambiguity
 2. Guidelines
- 3. Syntactic Requirements Patterns**

Textual Requirements Specification

Syntactic Requirements Patterns

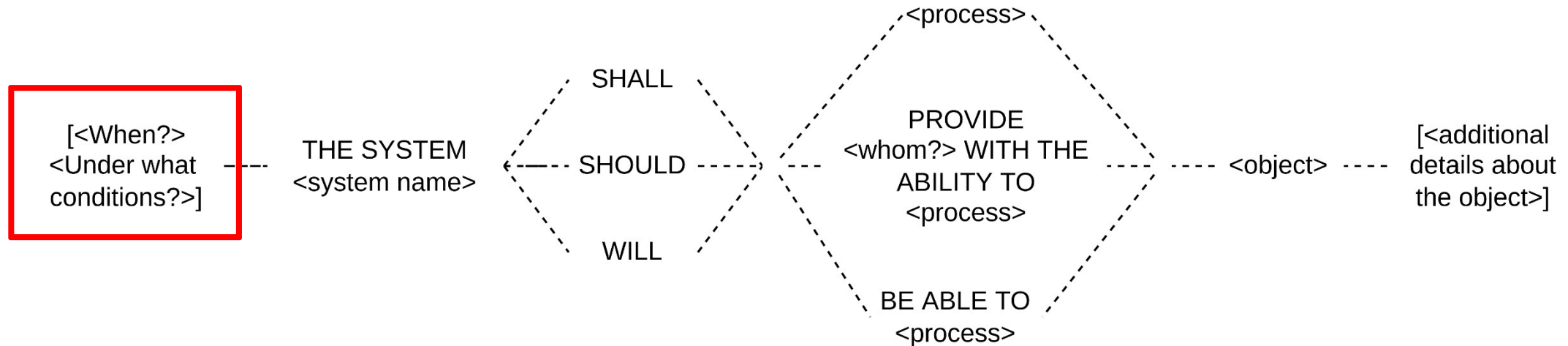
- Technique that aims at avoiding mistakes
- Also known as *requirement templates*

“A syntactic requirement pattern defines a syntactic structure for documenting requirements in natural language and defines meaning of each part of the syntactic structure.”

- A good pattern contains:
 - Condition, subject, “legal obligation”, verb, object

Textual Requirements Specification

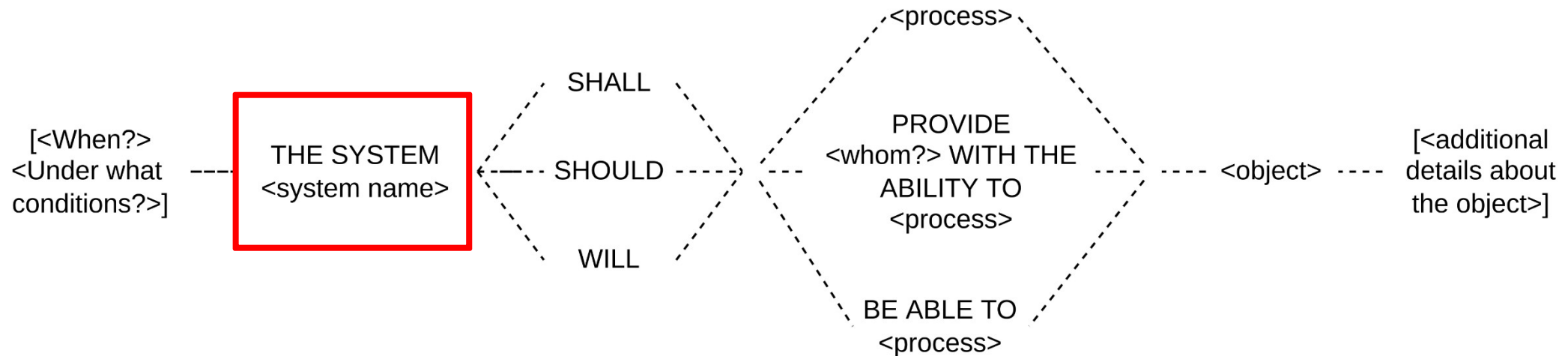
Syntactic Requirements Patterns - Example



- *<when?> / <under what conditions>*
 - Conditions under which the function documented in the requirement is performed
 - Temporal or logical
 - One or more

Textual Requirements Specification

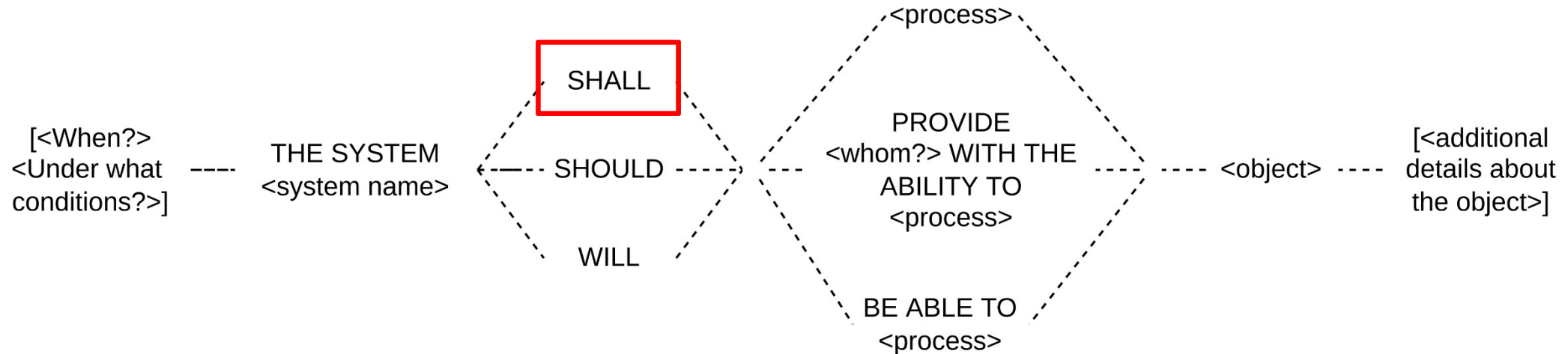
Syntactic Requirements Patterns - Example



- *THE SYSTEM / <system name>*
 - Name of the system that shall provide the functionality
 - Subject of the sentence

Textual Requirements Specification

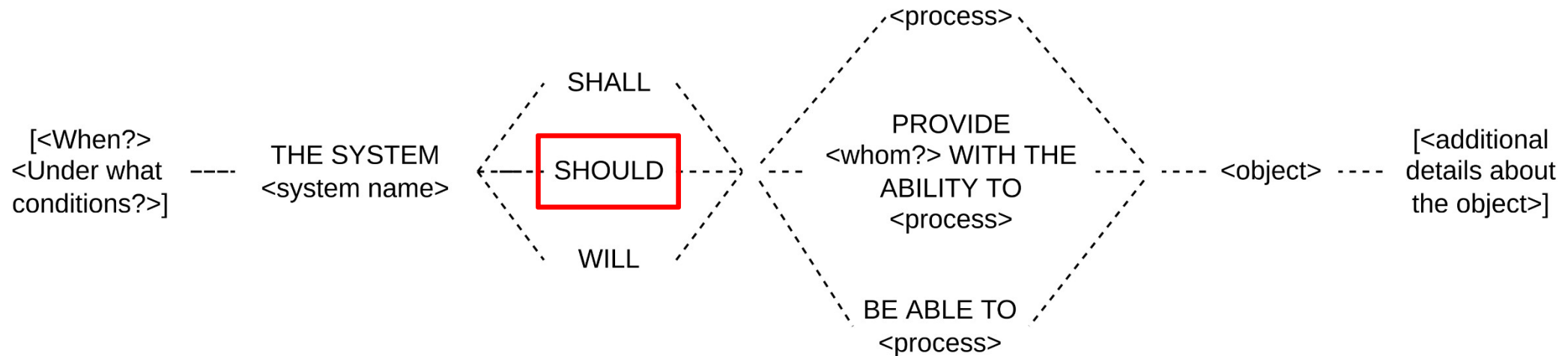
Syntactic Requirements Patterns - Example



- **SHALL**
 - Legally binding requirement
 - If a statement does not contain “shall”, it is not a requirement

Textual Requirements Specification

Syntactic Requirements Patterns - Example

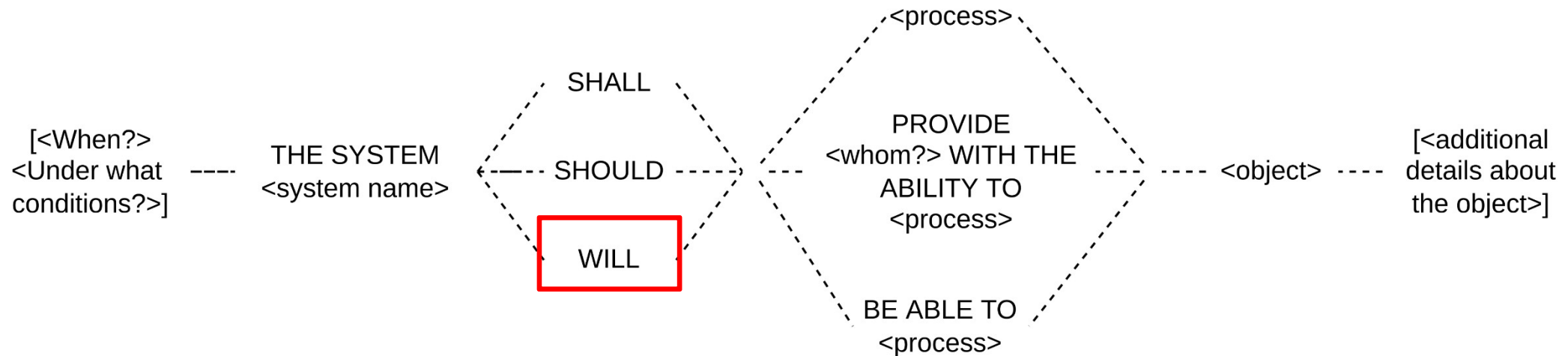


▪ *SHOULD*

- Highly recommended feature
- Optional, not contractually required
- More like goals instead of requirements

Textual Requirements Specification

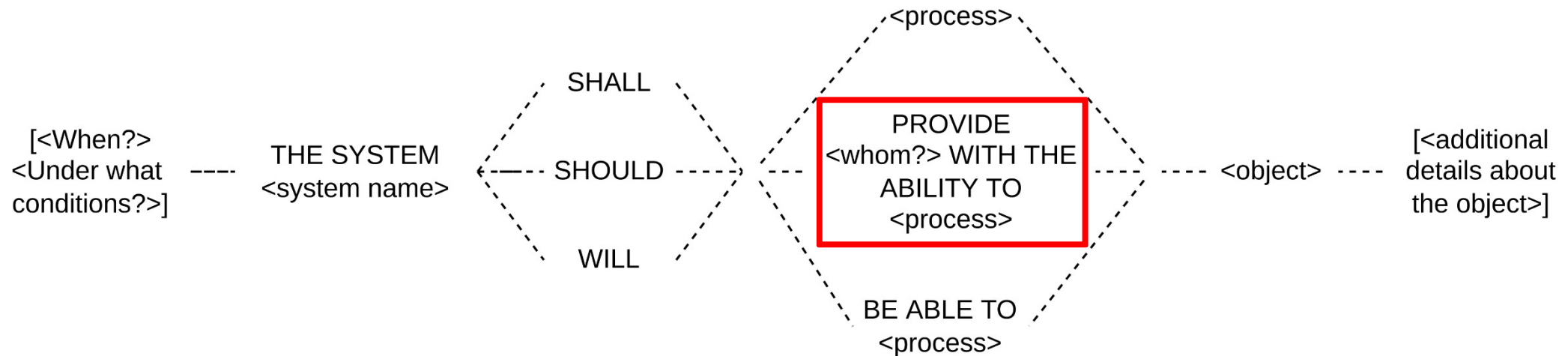
Syntactic Requirements Patterns - Example



- **WILL**
 - Statements of fact
 - Example: If I want to tell you something about another system I will use “will”.

Textual Requirements Specification

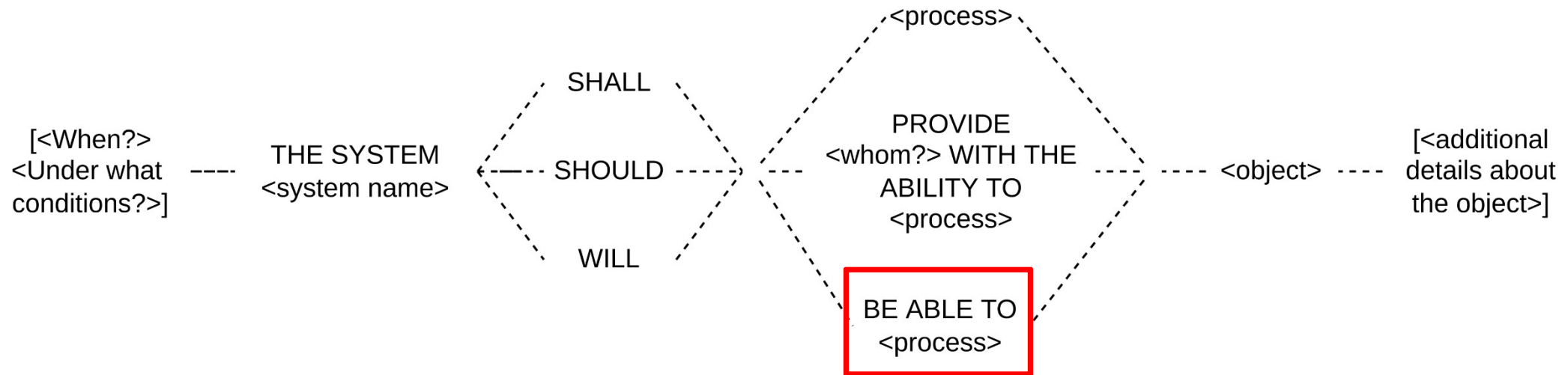
Syntactic Requirements Patterns - Example



- *PROVIDE <whom?> WITH THE ABILITY TO <process>*
 - Same as <process>, except: Applies to requirements offered to specific users → <whom?>

Textual Requirements Specification

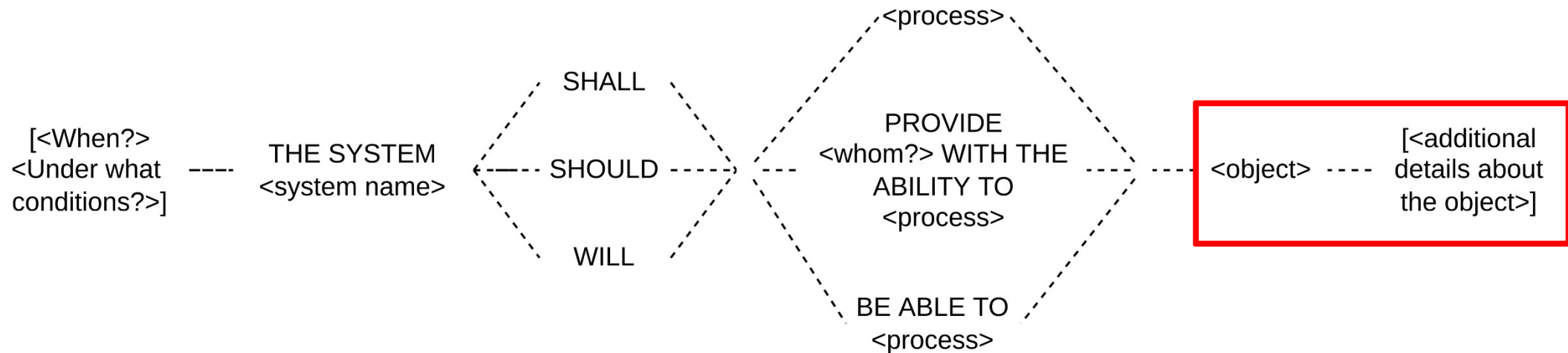
Syntactic Requirements Patterns - Example



- *BE ABLE TO <process>*
 - Same as <process>, except: Applies to requirements that are performed as reactions to trigger events from other systems

Textual Requirements Specification

Syntactic Requirements Patterns - Example



- *<object>* and *<additional details about the object>*
 - Object for which the functionality is required, e.g., which document shall be printed

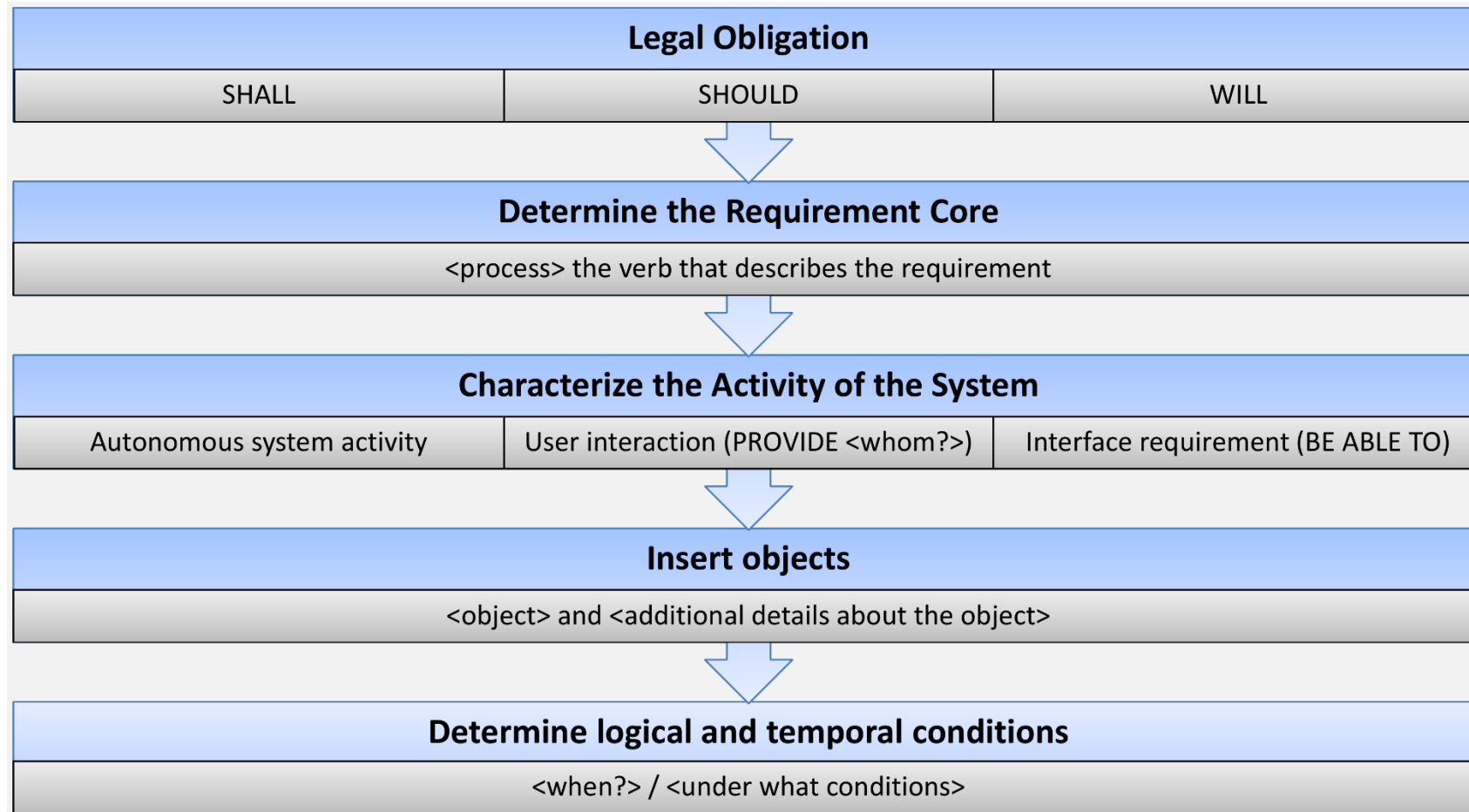
Textual Requirements Specification

Syntactic Requirements Patterns - Example

- “If the glass break detector detects the damaging of a window, the Burglar3000 shall inform the head office of the security service.”
 - <when>: if the glass break detector detects the damaging of a window
 - <system name>: the Burglar3000
 - SHALL
 - <process>: inform
 - <object>: the head office of the security service

Textual Requirements Specification

Syntactic Requirements Patterns - Fitting a Requirement into the Pattern



SUMMARY

Summary

- Requirements Documentation is a key artifact
 - Required amount of requirements documentation depends on context
- Natural language is a versatile means for requirements documentation
 - Versatility allows ambiguities and problems with the perspective
 - Ambiguity (multiple forms)
 - Guidelines for writing requirements documents
- Syntactic Requirements Patterns define a fixed structure for the requirements documentation
 - Condition, subject, legal obligation, verb, object



Questions?