A brief overview of requirements engineering

Professor Hossein Saiedian

EECS 448: Software Engineering

Fall 2022

KU THE UNIVERSITY OF KANSAS

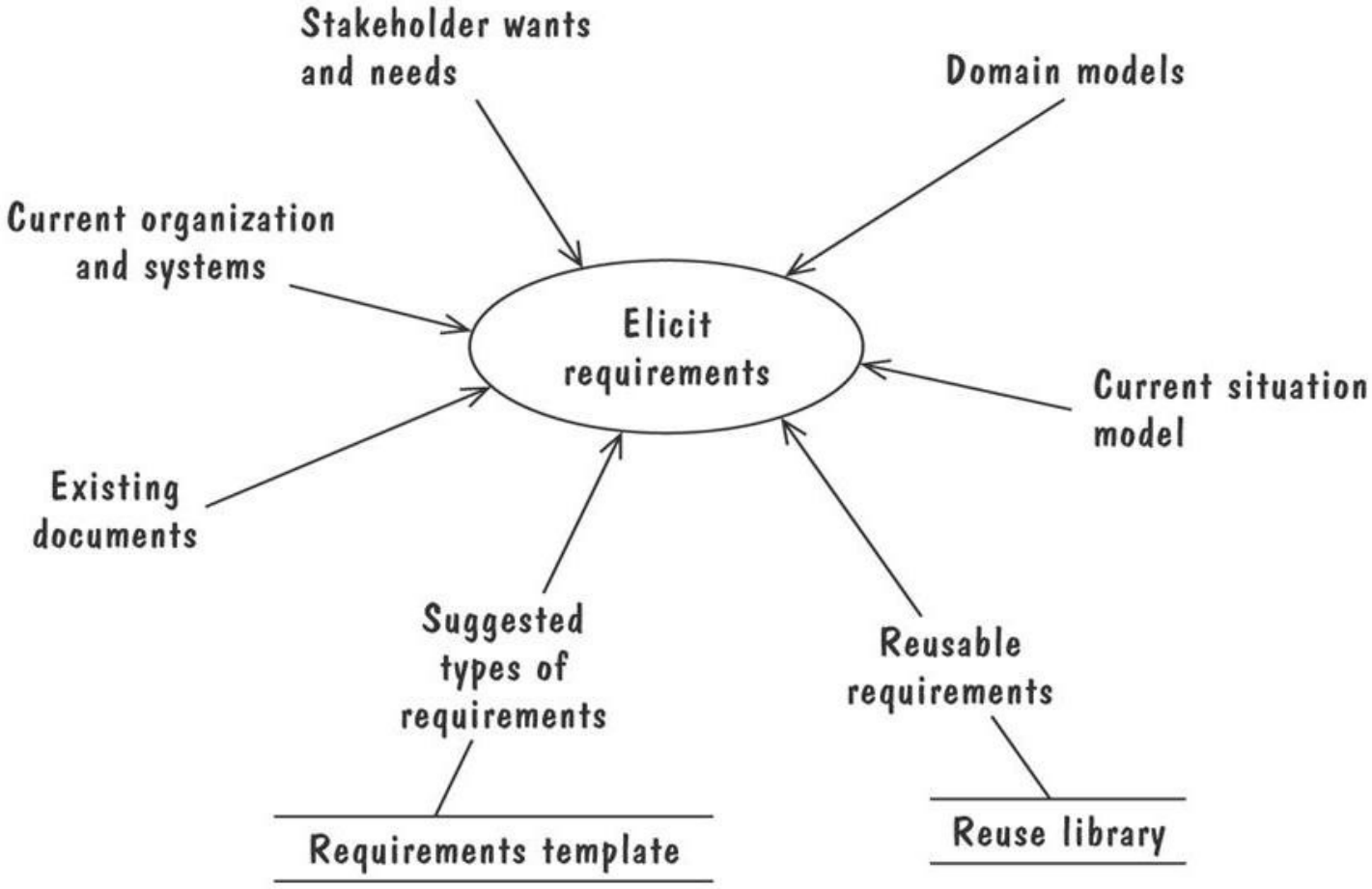
Requirements engineering

- A set that of activities that identify, document, and communicate the purpose of a new software
- A process that acts a bridge between the real-world needs of users and the capabilities afforded by the software
- The process of establishing the ***functionalities*** (services) expected from a new software and the ***constraints*** under which it operates and is developed
- Two important questions
  - **Why** a new system is needed, based on current or foreseen conditions,
  - **What** system features (services, functionalities) it will provide

What is a requirement

- A requirement is an expression of desired behavior
- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification
- Requirements are a specification of what should be implemented; they are descriptions of how the system should behave
  - Not how the software should be designed

Where do requirements come from



Common RE activities

- **Domain engineering**
- **Requirements elicitation**
  - Many different techniques
- **Requirements documentation**
  - Many different forms and notations
- **Requirements validation**

Domain engineering

- Before software is developed, we must understand the requirements
- Before requirements can be finalized, we must understand the domain
- What do we mean by a domain?
  - An area of natural or human activity
    * Healthcare, railways, banking, aerospace, chemical engineering, stocks
- Other terms
  - Domain analysis, domain understanding, domain modeling

Why domain engineering

- It is knowledge acquisition
- The objective is to learn
  - About the organization: its structure, business needs, culture, roles, responsibilities, stakeholders
  - About the domain: concepts, terminologies, regulations

Why domain engineering

- Better prep/understanding current and future apps that share a set of common characteristics
  - Good understanding of a domain is key to identifying and developing reusable software components
    * Establish a *product line*
    * A software product line is a set of software systems sharing a common, managed set of features that satisfy the specific needs of a particular domain and are developed from a common set of core assets

Examples of domains

- Financial industry: banking, insurance, securities, …
- Healthcare: hospitals, clinics, patients, doctor offices, …
- Transportations: road, railway, sea, airways, …
- Oil and gas systems: pumps, pipes, valves, refineries, distribution
- …

Example of a specific domain

- Railway
  - Tracks, lines, platforms, switch, crossover, siding, stations, rails
  - Rail: length, topology, context (in a tunnel, along a platform, …)
- An understanding of all important concepts is most essential
  - Can be presented informally and formally

An informal description of railway nets

- A railway net consists of one or more lines and two or more stations.
- A railway net consists of rail units.
- A line is a linear sequence of one or more linear rail units.
- The rail units of a line must be rail units of the railway net of the line.
- A station is a set of one or more rail units.
- The rail units of a station must be rail units of the railway net of the station.
- No two distinct lines and/or stations of a railway net share rail units.
- …

A formal description of railway nets

1. A railway net consists of one or more lines and two or more stations.
2. A railway net consists of rail units.

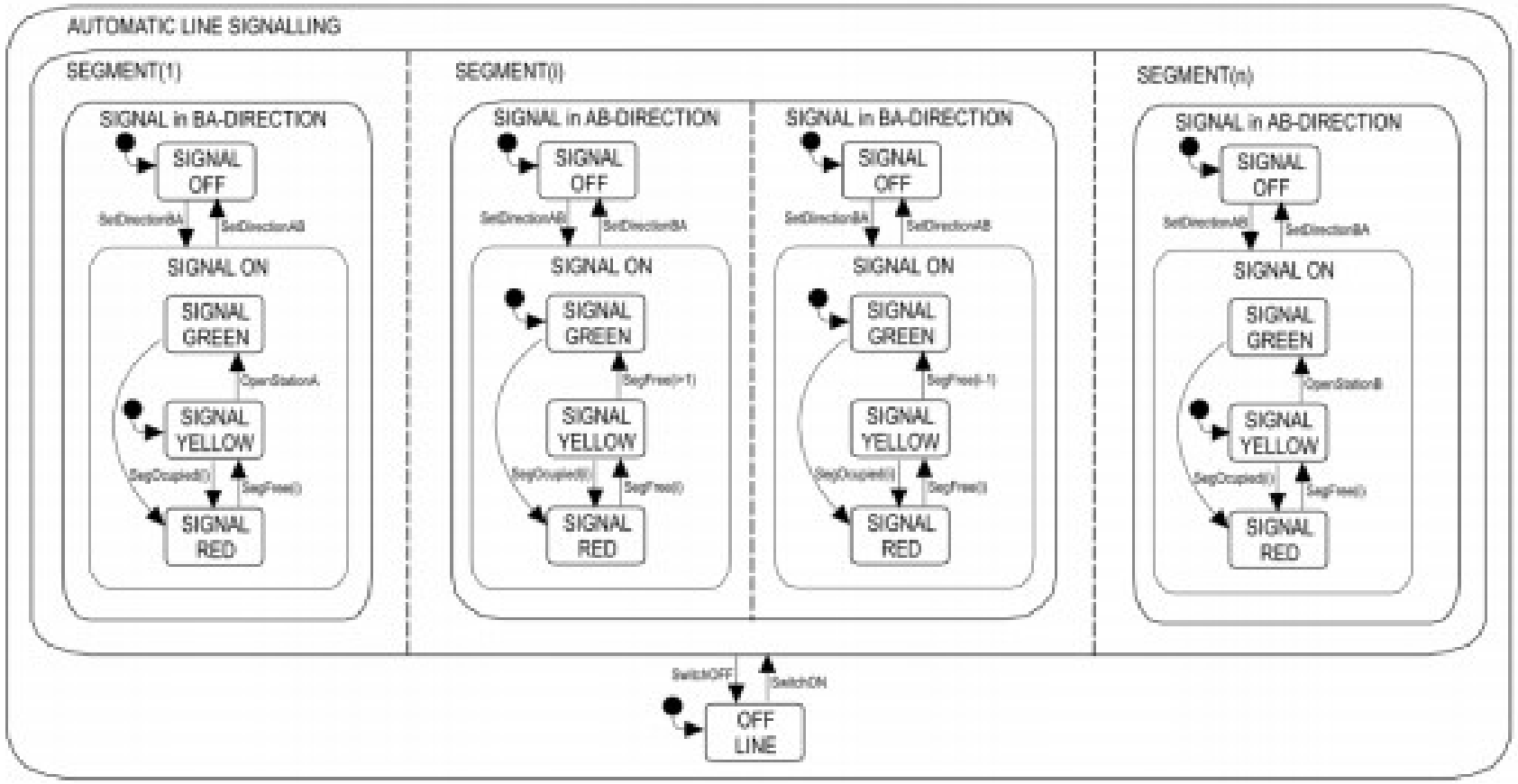
value

1. obs_Ls : N → L-**set**
1. obs_Ss : N → S-**set**
2. obs_Us : N → U-**set**

axiom

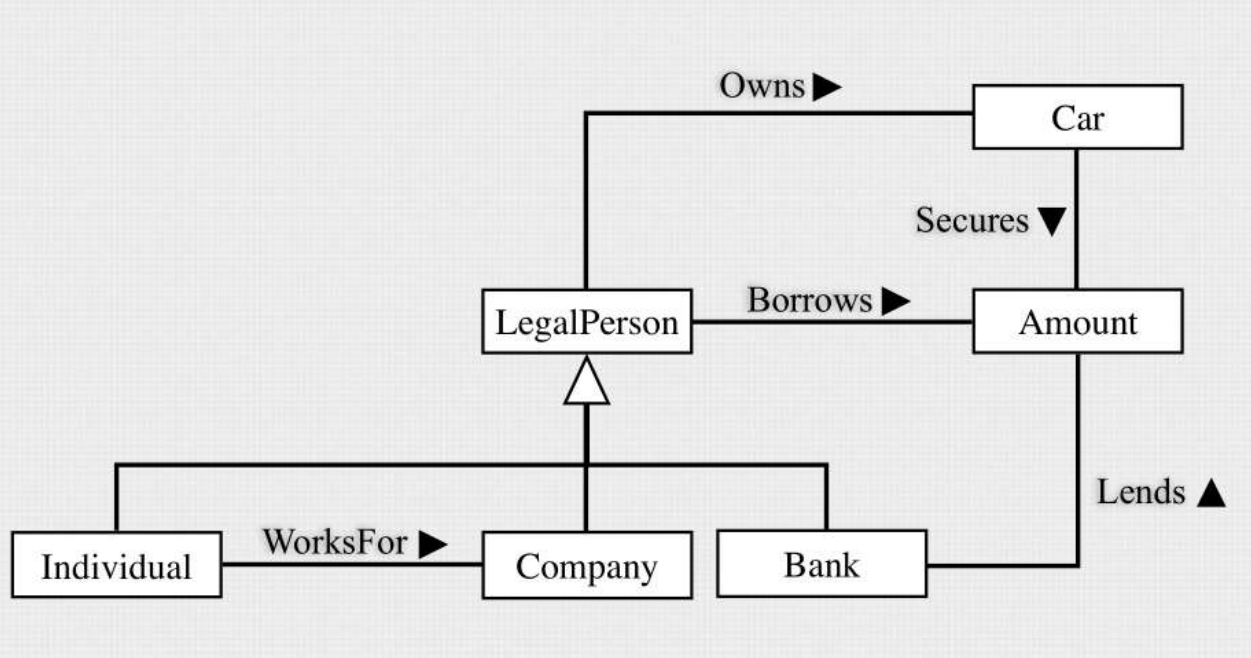
1. ∀ n:N • **card** obs_Ls(n) ≥ 1
1. ∀ n:N • **card** obs_Ss(n) ≥ 2

Visual description of railway nets



Sketching a domain model

- A person can work for one or several companies.
- A car is owned by a person, a bank, or a company.
- Banks give loans for buying cars.
- A loan can be secured against a car.



Summary

- The Domain Model captures the concepts in the domain of the problem, and the relationships between them.
- It establishes the vocabulary of the problem domain.

Common RE activities

- Domain engineering
- **Requirements elicitation**
  - Many different techniques
- Requirements documentation
  - Many different forms and notations
- Requirements validation

Requirements elicitation

- Objective: to identify the key stakeholders, and to communicate and collaborate with them to identify and document a software product features
- **Artifact-driven**: rely on the specific types of artifacts (documents) to obtain knowledge
- **Stakeholder-driven**: rely on specific types of interaction with stakeholders to obtain knowledge

Artifact-driven

- Rely on the specific types of artifacts to obtain knowledge
  - Background study
  - Data collection, survey questionnaires
  - Reports
  - Scenarios, storyboards for problem world exploration
  - Prototypes, mock-ups for early feedback
  - Knowledge reuse for domain-specific software
  - …

Stakeholder-driven

- Rely on specific types of interaction with stakeholders to obtain knowledge
  - Interviews
  - Observation and ethnographic studies
  - Group sessions
  - …
- Important to identify important stakeholders
  - Essential for building a shared understanding of problems
  - Critical to obtaining complete, adequate and realistic requirements
  - Analysis of stakeholders should be based on their respective role, interests and type of knowledge they can contributes

Stakeholder-driven

- Important to identify important stakeholders
  - Essential for building a shared understanding of problems
  - Critical to obtaining complete, adequate and realistic requirements
  - Analysis of stakeholders should be based on their respective role, interests and type of knowledge they can contributes

Types of requirements

- Functional requirements
- Non-functional requirements (AKA quality attributes)
- Constraints

Types of requirements

- Functional requirements
  - Describe what the system should do
  - Address the "WHAT" aspects
  - Functional requirements characterize units of functionality that are sometime called features i
  - Example: the train control software shall control the acceleration of all the system's trains

Non-functional requirements

- Non-functional requirements: Define constraints on the way the software should satisfy its functional requirements
  - Performance, security, portability, interoperability, …
- Constraints
  - Sometimes documented separately, sometimes documented as part of the non-functional requirements
  - Constraints about the development process
    * Use the spiral model, use C++ for programming, …

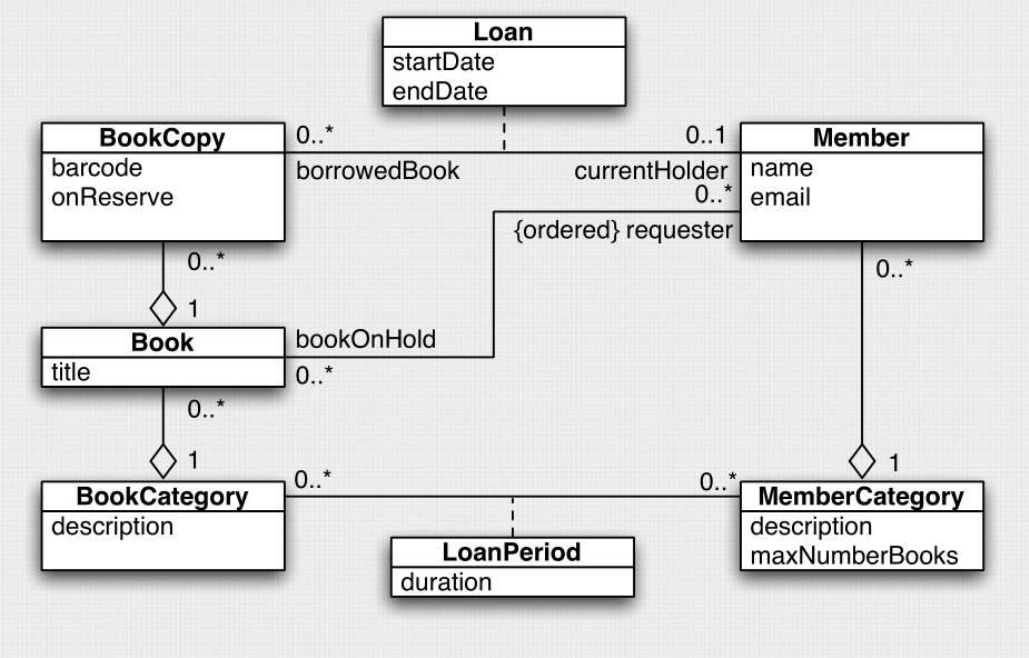
Common RE activities

- Domain engineering
- Requirements elicitation
  - Many different techniques
- **Requirements documentation**
  - Many different forms and notations
- Requirements validation

Requirements documentations

- Consists of detailing, structuring and documenting the characteristics of the new software
- Common artifacts
  - Software requirements definitions normally in an easy-to-understand language for the customers/users
  - Requirement models
    * Many different notations; will use UML *use case* modeling
  - Software requirement specification (SRS)

Examples of modeling requirements



LibSystem

$$members : \mathbb{P}\, PERSON$$
$$shelved : \mathbb{P}\, BOOK$$
$$checked : BOOK \pfun PERSON$$

$$shelved \cap \operatorname{dom} checked = \varnothing$$
$$\operatorname{ran} checked \subseteq members$$
$$\forall\, mem : PERSON \bullet \#(checked \rhd \{mem\}) \leq MaxLoan$$

Requirement validation

- Objective: quality assurance
- The specifications should be *verified* against each other in order to find inconsistencies and omissions
- The specifications should be *validated* with stakeholders in order to pinpoint inadequacies with respect to actual needs

VERIFICATION

- 2 sleeves?
- Is it size L?
- Is it blue?
- Are any buttons missing?

VALIDATION

- Does it fit?
- Is it comfortable to drive in?
- Does the colour match my eyes?
- Can I afford it?
- Is it good quality?
- Will my date like it?

Summary: Characteristics of good requirements

- *Completeness*: the requirements, assumptions and domain properties must be sufficient to ensure that the new system will satisfy its expected objectives
- *Consistency*: must be compatible with each other
- *Unambiguity*: the requirements, assumptions and domain properties must be formulated in way that precludes different interpretations

Summary: Characteristics of good requirements

- *Good structuring*: organized in a way that highlights the structural links among its elements
- *Modifiability*: should be possible to revise, adapt, extend or contract the requirements document through modifications that are as local as possible
- *Traceability*: the context in which an item of the RD was created, modified or used should be easy to retrieve

Summary: Characteristics of good requirements

- *Pertinence*: the requirements and assumptions must all contribute to the satisfaction of one or several objectives underpinning the system-to-be
- *Feasibility*: requirements must be realizable in view of the budget, schedule and technology constraints
- *Comprehensibility*: formulation of requirements, assumptions and domain properties must be comprehensible by the people who need to use them