

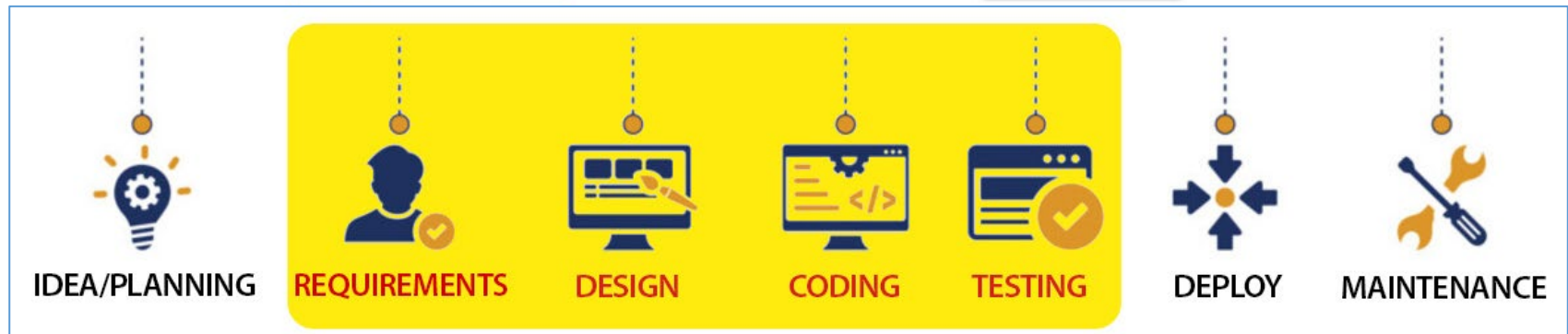
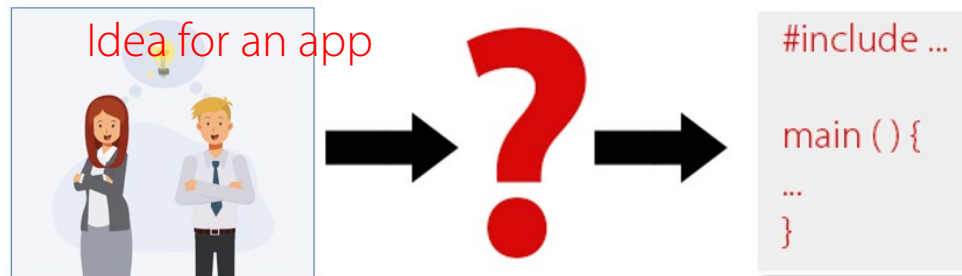
# *Software requirements engineering*

**Professor Hossein Saiedian**

EECS 348: Software Engineering

Spring 2024

- We focus on the *development* life cycle
  - Requirements engineering
  - Design
  - Construction (implementation, coding)
  - Testing





A set that of activities that identify, document, and communicate the purpose of a new software

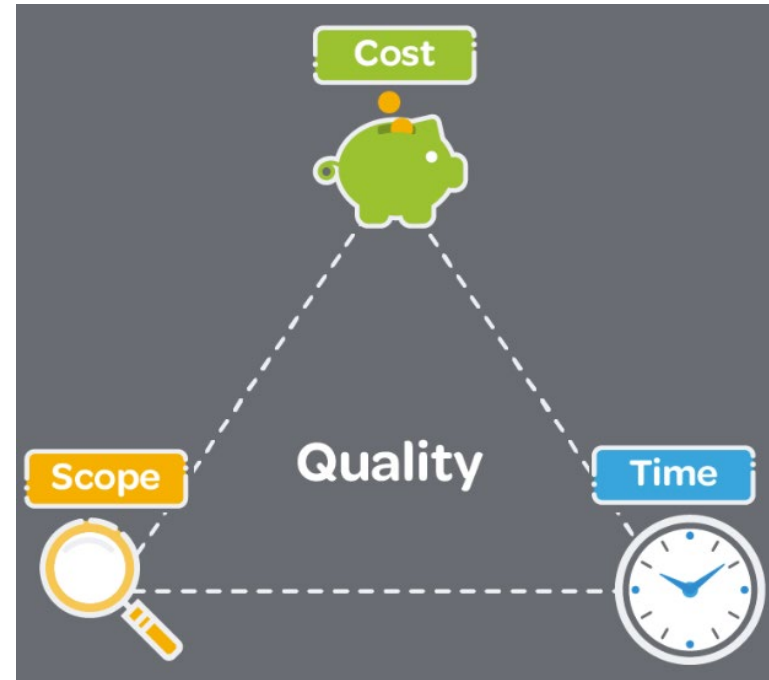


A process that acts a bridge between the real-world needs of users and the capabilities (and opportunities) afforded by the software



The process of establishing the services that are expected from a system 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
- A software project is successful if the system satisfies its software requirements, the budget is not overrun, and the system is delivered as scheduled
  - Budget
  - Scope
  - Schedule
- Identifying the real requirements is essential for the success of a project



# What is a requirement

- A requirement is an expression of desired behavior
- Software requirements are capabilities that the system must deliver
- 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*

- Stakeholders needs
- Domain model
- Requirement templates (suggested requirements)
- Current situation model
  - How things are currently being done
  - Documents and manuals



**Domain engineering**



**Requirements elicitation**



**Requirements modeling  
and documentation**



**Requirements validation**



**Domain engineering**



**Requirements elicitation**



**Requirements modeling  
and documentation**



**Requirements validation**



- 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

- It is all about learning about an environment
- The objective is to learn
  - About the organization: its structure, business needs, culture, roles, responsibilities, stakeholders
  - About the domain: concepts, terminologies, regulations

- **Financial industry:** banking, insurance, securities, ...
- **Healthcare:** hospitals, clinics, patients, doctor offices, ...
- **Transportations:** airways, railway, highways, cruises
- **Oil and gas** systems: pumps, pipes, valves, refineries, distribution
- ...

- 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

- 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.
- ...

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

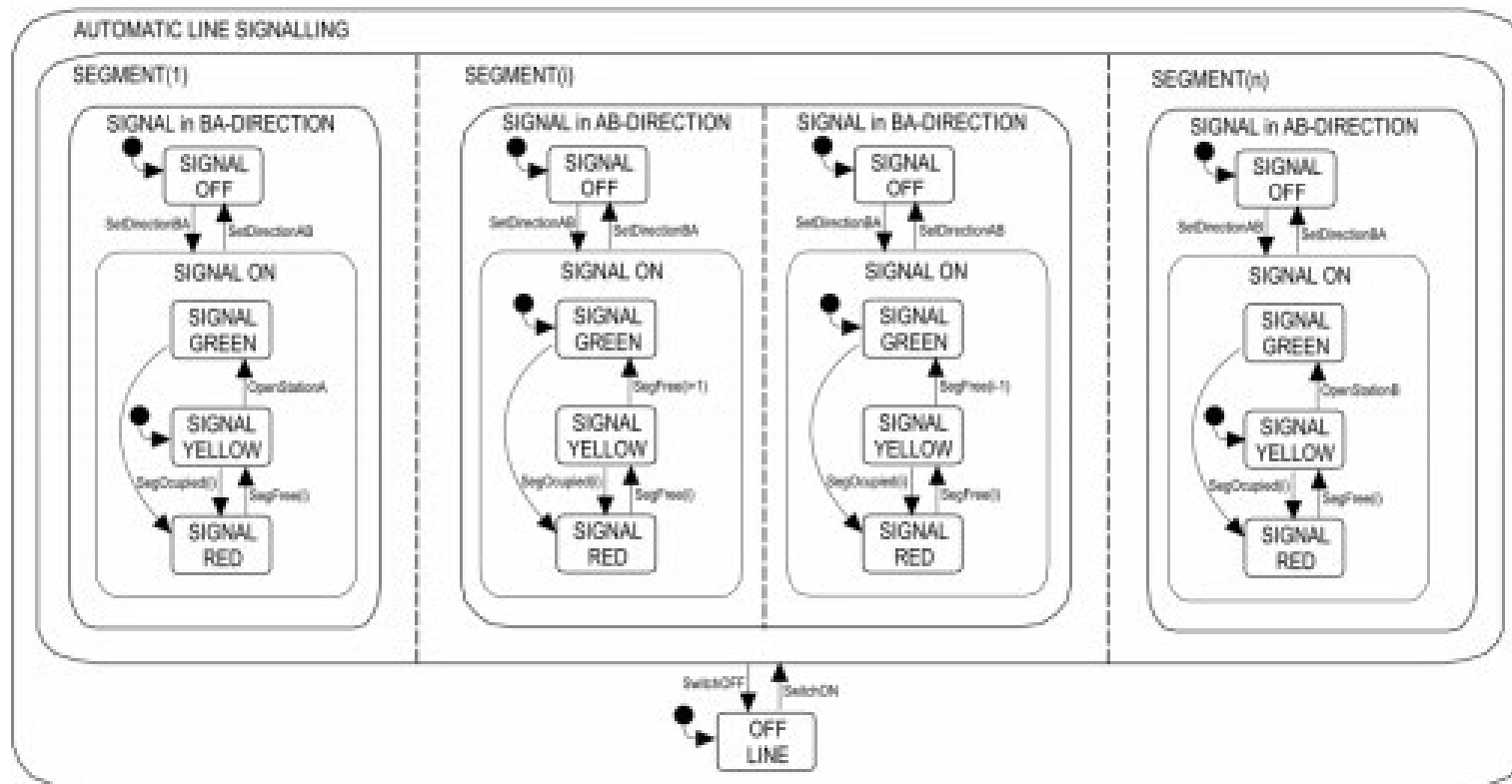
## value

1.  $\text{obs\_Ls} : N \rightarrow \text{L-set}$
1.  $\text{obs\_Ss} : N \rightarrow \text{S-set}$
2.  $\text{obs\_Us} : N \rightarrow \text{U-set}$

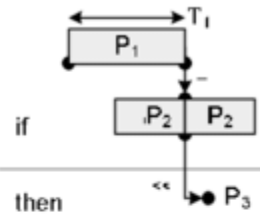
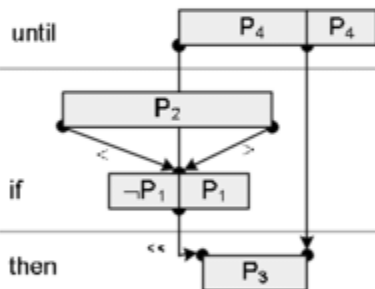
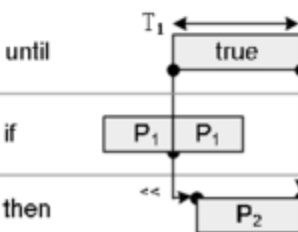
## axiom

1.  $\forall n:N \bullet \text{card } \text{obs\_Ls}(n) \geq 1$
1.  $\forall n:N \bullet \text{card } \text{obs\_Ss}(n) \geq 2$

# Visual description of railway nets



# Examples of modeling requirements

	Prosa Requirement	Requirement Sentence Template	In FRL $R = \left( \begin{smallmatrix} \text{then} \\ A \rightarrow B \rightarrow C \\ \text{first} \end{smallmatrix} \right)$	Graphical
R <sub>1</sub>	If button B4 has not been down during the last 5 seconds, then lamp L3 must be lit immediately after button B4 changed to down.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is not down</span> P <sub>1</sub> held the last <span style="border: 1px solid black; padding: 2px;">5 seconds</span> T <sub>1</sub> and now <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>2</sub> occurs, then must <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is lit</span> P <sub>3</sub> follow immediately.	$A = [P_1]_{t_1}^{t_2} \wedge [P_2]_{t_3}^4 \wedge (t_2   t_3) \wedge (t_1 = t_2 - T_1)$ $B = [P_3]_{t_5}^{t_6} \wedge (t_3   t_5)$	
R <sub>2</sub>	Immediately after button B4 is released while the system is in mode m1 and lamp L3 is not lit, the lamp L3 must be lit until button B4 is down again or the system leaves m1.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>1</sub> occurs, during <span style="border: 1px solid black; padding: 2px;">System is in mode m1 AND</span> <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is not lit</span> P <sub>2</sub> holds, then <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is lit</span> P <sub>3</sub> must hold immediately, until <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is up</span> OR <span style="border: 1px solid black; padding: 2px;">System is not in mode m1</span> P <sub>4</sub> hold.	$A = [[P_1]_{t_1}^{t_2} \wedge [P_2]_{t_3}^4 \wedge (t_3 < t_1 < t_4)$ $B = [P_3]_{t_5}^{t_6} \wedge (t_1   t_5)$ $C = [P_4]_{t_7}^{t_8} \wedge (t_6   t_7)$	
R <sub>3</sub>	Immediately after button B4 is pressed, Lamp L3 must be red for 10 seconds.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>1</sub> occurs, then <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is red</span> P <sub>2</sub> hold immediately, until <span style="border: 1px solid black; padding: 2px;">10 seconds</span> T <sub>1</sub> elapsed.	$A = [[P_1]_{t_1}^{t_2}$ $B = [P_2]_{t_3}^{t_4} \wedge (t_2   t_3)$ $C = (t_4 = t_3 + 10s)$	



# Example 2: A study abroad system

- Study abroad management system (Kung, SE)

“An undergraduate student or a graduate student can apply to no more than two exchange programs per semester. An application consists of an application form, two faculty recommendation letters, and a course equivalency form to be approved by an academic advisor.

...

An exchange program has a program name, program type, academic department, academic subject, country, region, term of study, and language.”

# Example 2: A study abroad system

- Study abroad management system (Kung, SE)

“An undergraduate student or a graduate student can apply to no more than two exchange programs per semester. An application consists of an application form, two faculty recommendation letters, and a course equivalency form to be approved by an academic advisor.

...

An exchange program has a program name, program type, academic department, academic subject, country, region, term of study, and language.”



An undergraduate student<sup>1</sup> or a graduate student<sup>1</sup> can apply to<sup>3</sup> no more than two<sup>5</sup> exchange programs<sup>1</sup> per semester<sup>1</sup>. An application<sup>1</sup> consists of<sup>7</sup> an application form<sup>1</sup>, two<sup>5</sup> faculty recommendation letters<sup>1</sup>, and a course equivalency form<sup>1</sup> to be approved by<sup>3</sup> an academic advisor<sup>1</sup>.

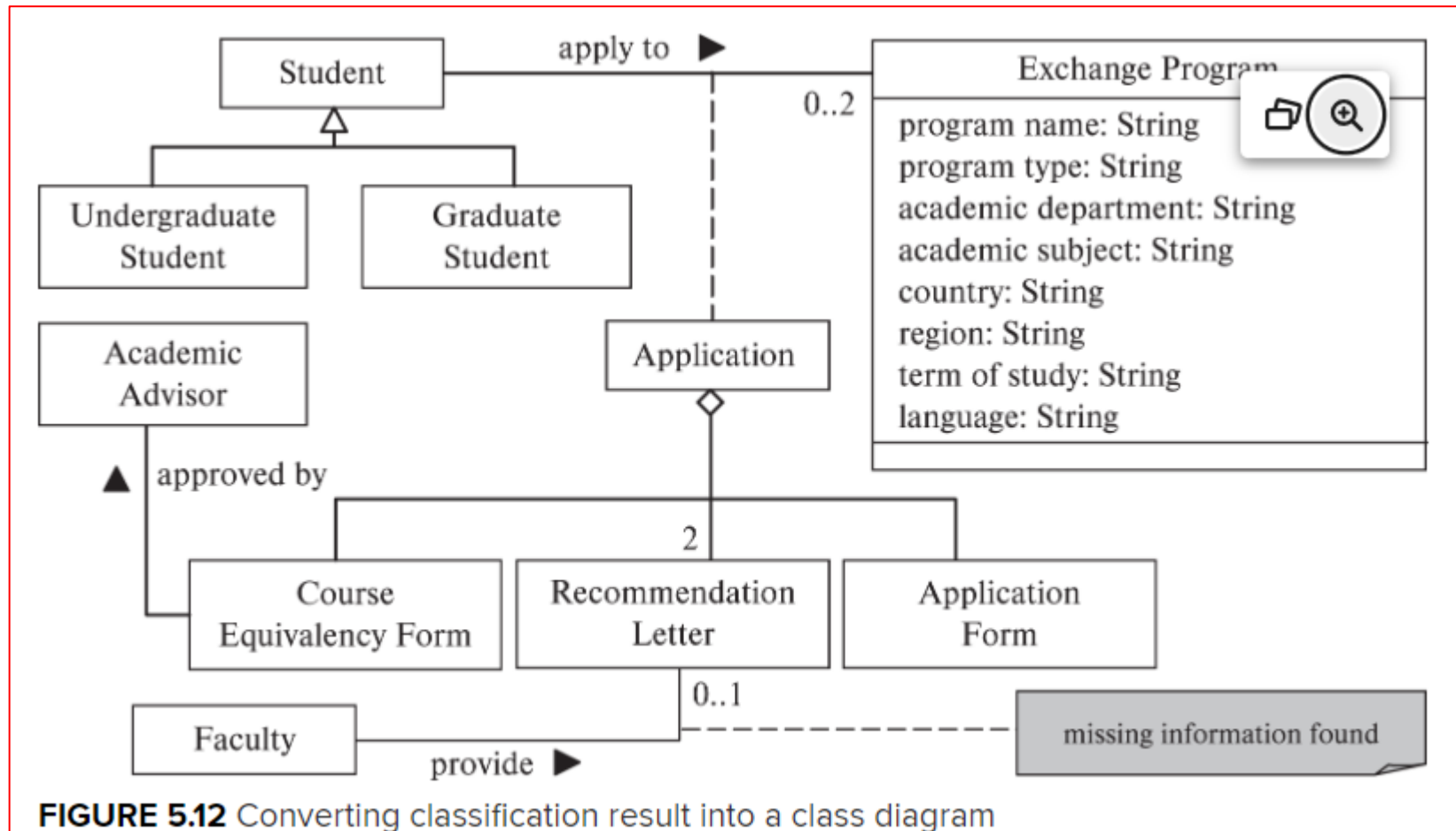
...

An exchange program has<sup>6</sup> a program name<sup>1</sup>, program type<sup>1</sup>, academic department<sup>1</sup>, academic subject<sup>1</sup>, country<sup>1</sup>, region<sup>1</sup>, term of study<sup>1</sup>, and language<sup>1</sup>.

undergraduate student<sup>1</sup>  
graduate student<sup>1</sup>  
apply to<sup>3</sup>  
no more than two<sup>5</sup>  
exchange programs<sup>1</sup>  
semester<sup>1</sup>  
application<sup>1</sup>  
consists of<sup>7</sup>  
application form<sup>1</sup>  
two<sup>5</sup>  
faculty<sup>1</sup>  
faculty  
recommendation letters<sup>1</sup>  
course equivalency form<sup>1</sup>  
approved by<sup>3</sup>  
academic advisor<sup>1</sup>  
has<sup>6</sup>

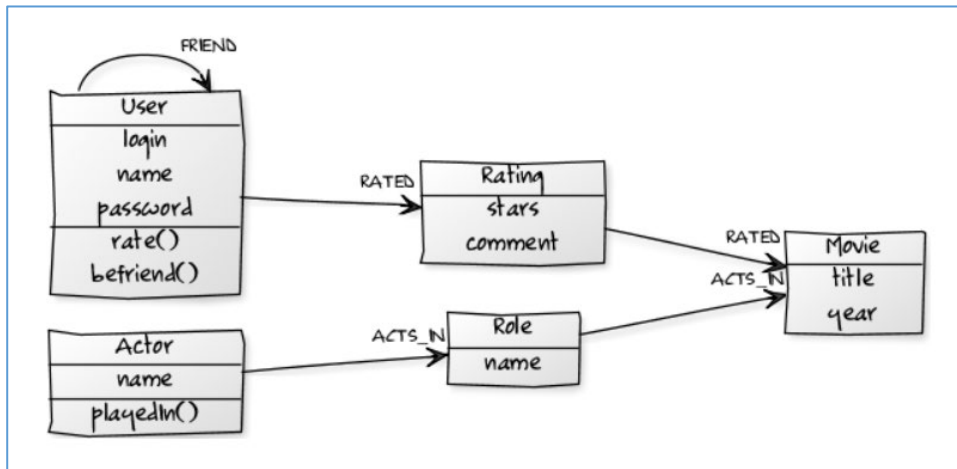
# Example 2: A study abroad system

- Study abroad management system (Kung, SE)



# A domain model: another example

- Another example
  - Domain model for **themoviedb.org**
  - Notation: UML



```
class Movie {
    String id;
    String title;
    int year;
    Set<Role> cast;
}

class Actor {
    String id;
    String name;
    Set<Movie> filmography;
    Role playedIn(Movie movie, String role) { ... }
}

class Role {
    Movie movie;
    Actor actor;
    String role;
}

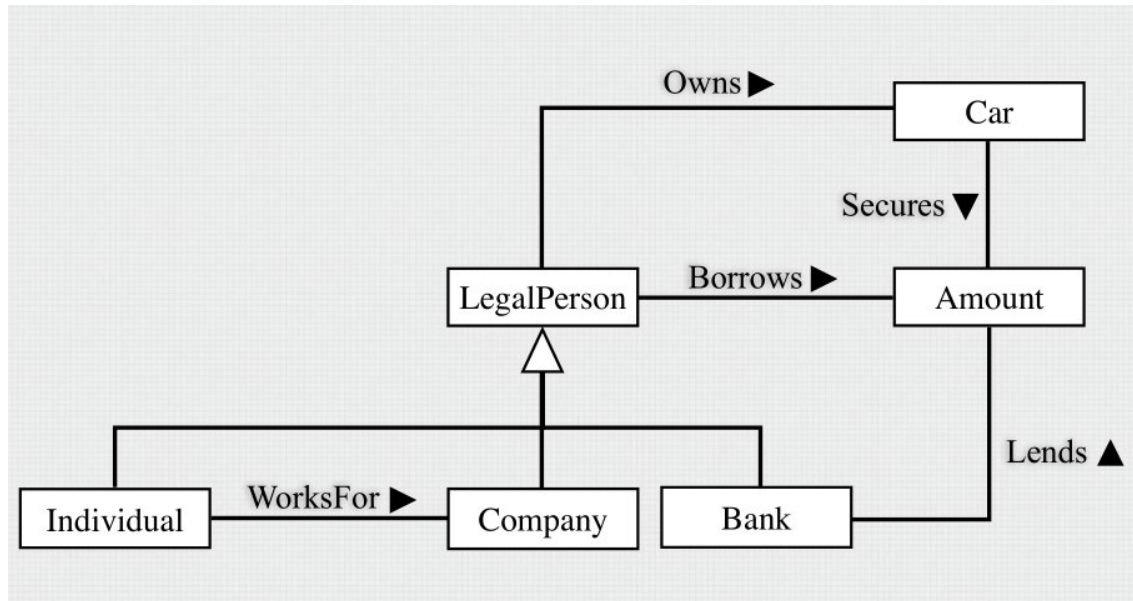
class User {
    String login;
    String name;
    String password;
    Set<Rating> ratings;
    Set<User> friends;
    Rating rate(Movie movie, int stars, String comment) { ... }
    void befriend(User user) { ... }
}

class Rating {
    User user;
    Movie movie;
    int stars;
    String comment;
}
```

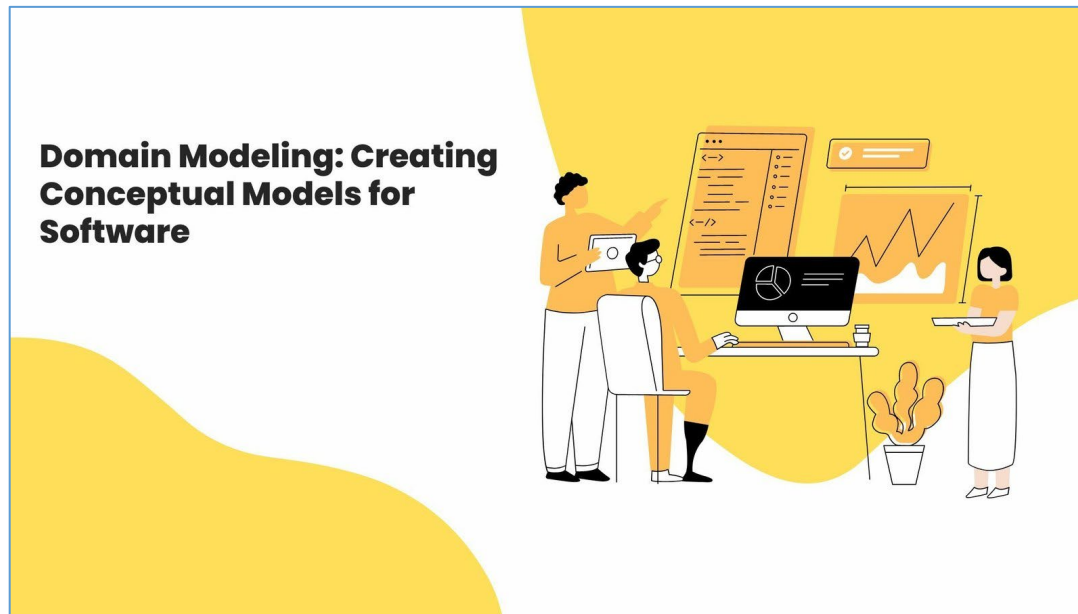
[https://docs.spring.io/spring-data/data-neo4j/docs/3.1.0.RELEASE/reference/html/tutorial\\_domain.html](https://docs.spring.io/spring-data/data-neo4j/docs/3.1.0.RELEASE/reference/html/tutorial_domain.html)

# A domain model: another example

- 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.



- A domain model captures the concepts in the domain of the problem and the relationship between them
- It establishes the vocabulary of the problem domain





**Domain engineering**



**Requirements elicitation**



**Requirements modeling  
and documentation**



**Requirements validation**

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



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

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

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

- Functional requirements
  - Describe what the system should do
  - Address the **what** aspects
  - Functional requirements characterize units of functionality that are sometime called features
  - Examples

**1. Train Scheduling:**

- The system should allow administrators to create and modify train schedules.
- It should enable automated allocation of time slots for each train based on predefined criteria such as availability of tracks and station capacity.

**2. Ticket Reservation System:**

- Users should be able to search for available trains based on criteria such as departure and destination stations, date, and time.
- The system should facilitate online booking and cancellation of tickets.
- It should provide real-time updates on seat availability.

- Functional requirements: another example (airline app)

## 1. Flight Search and Booking:

- The system must allow users to search for flights based on criteria such as departure and arrival locations, dates, and class of service.
- Users should be able to view available flights, select a flight, and book tickets.

## 2. User Account Management:

- The system must enable users to create and manage their accounts.
- Users should be able to update personal information, view booking history, and manage preferences.

## 3. Payment Processing:

- The system must support secure payment processing for booking flights.
- It should accept multiple payment methods, including credit/debit cards, digital wallets, and bank transfers.

## 4. Reservation Management:

- The system must allow users to view, modify, or cancel their reservations.

- Functional requirements: another example (a library system)

## 1. Catalog Management:

- The system must allow librarians to add, update, and delete book records.
- Users should be able to search the catalog by title, author, genre, and ISBN.

## 2. User Account Management:

- The system must enable users to create and manage their library accounts.
- Users should be able to update personal information, view borrowing history, and manage their book reservations.

## 3. Borrowing and Returning Books:

- The system must allow users to borrow and return books.
- It should track due dates, send reminders for overdue books, and calculate any applicable fines.

## 4. Reservation and Hold Requests:

- The system must allow users to place holds on books that are currently checked out.

- Non-functional requirements: Define constraints on the way the software should satisfy its functional requirements
  - Performance
  - Security
  - Portability
  - Interoperability
  - Reliability
  - ...

- Non-functional requirements for airline apps

- Performance

**Performance:**

- The system must handle up to 10,000 concurrent users during peak times without significant degradation in response time, ensuring that search results and booking confirmations are processed within 2 seconds.

- Security

**Data Encryption:** All sensitive data, including personal passenger information and payment details, must be encrypted both in transit and at rest using industry-standard encryption protocols (e.g., AES-256). This ensures that even if data is intercepted or accessed without authorization, it remains unreadable and secure.



- Non-functional requirements for airline apps

- Portability

**Platform Independence:** The system must be designed to operate seamlessly across various operating systems (e.g., Windows, macOS, Linux) and devices (e.g., desktops, tablets, smartphones) without requiring significant modifications. This ensures that users can access the system from different environments with consistent performance and functionality.

- Interoperability

**Interoperability:** The system must be capable of seamlessly exchanging data with other external systems, such as partner airlines, travel agencies, and payment gateways, using standardized communication protocols (e.g., RESTful APIs, SOAP). This ensures smooth and efficient data flow between different systems, enhancing the overall user experience and operational efficiency.

- Reliability

**Reliability:** The system must achieve an uptime of at least 99.99% over a 12-month period, ensuring continuous availability and minimal downtime. Additionally, it should include robust failover mechanisms and regular automated backups to prevent data loss and ensure quick recovery in case of system failures.

- Constraints
  - Sometimes documented separately, sometimes documented as part of the non-functional requirements
  - Constraints about the development process
    - \* Use C++ for programming, develop on a Linux environment, ...
  - An example

## Development Constraint

**Programming Language Requirement:** The airline reservation system must be developed using **Java**. This constraint ensures robust performance and scalability, which are critical for handling high volumes of transactions and user requests. Java's extensive libraries and frameworks also support strong security features and facilitate integration with other systems, such as payment gateways and flight management systems. Additionally, using Java aligns with the long-term maintenance and support strategy, given its widespread use and the availability of skilled developers.

- Assume a medical application
  - The application must allow users to record and view patient medical histories.
  - The application must allow users to generate reports on patient data.
  - The application must allow users to order medications for patients.
  - The application must allow users to schedule appointments for patients.
  - The application must allow users to communicate with other healthcare providers about patients.



- Assume the same medical application
  - The application must be secure and protect patient data.
  - The application must be reliable and available 24/7.
  - The application must be easy to use and navigate.
  - The application must be scalable to support many users.
  - The application must meet all applicable regulations.

- Assume the same medical application
  - The application must be developed using open-source software.
  - The application must be hosted on-premises.
  - The application must be completed within 6 months.
  - The application must be within the budget of \$1 million.
  - The application must be compatible with the existing healthcare system.

# Requirements elicitation: summary



client presentation



existing documents



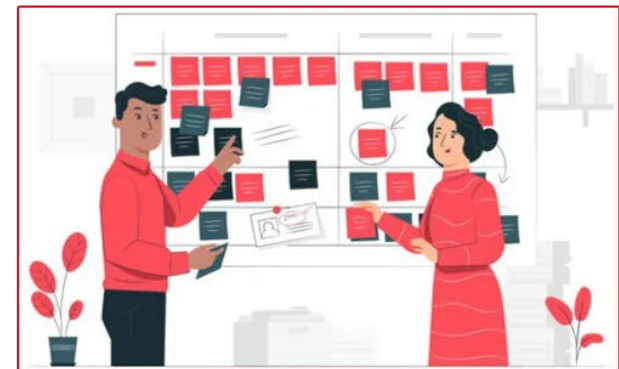
stakeholders interview



surveys



standards and templates



user stories



**Domain engineering**



**Requirements elicitation**



**Requirements modeling  
and documentation**

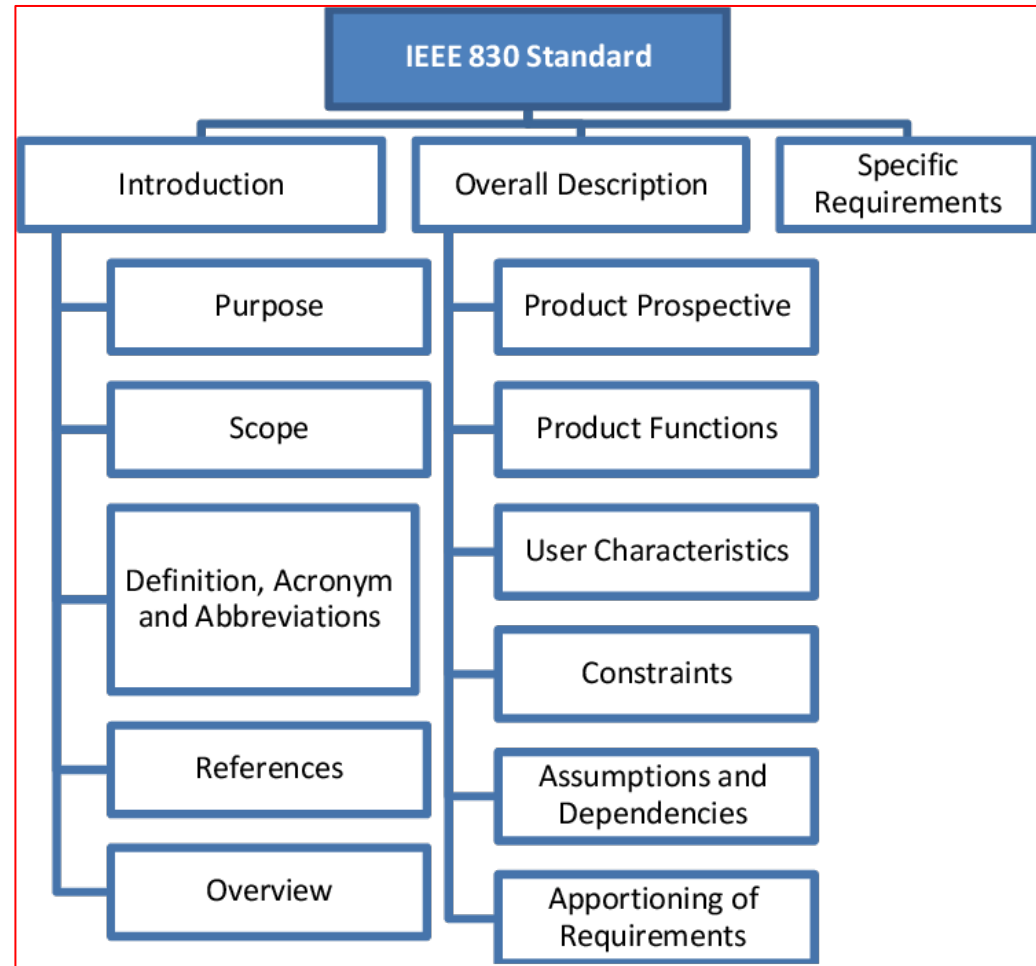
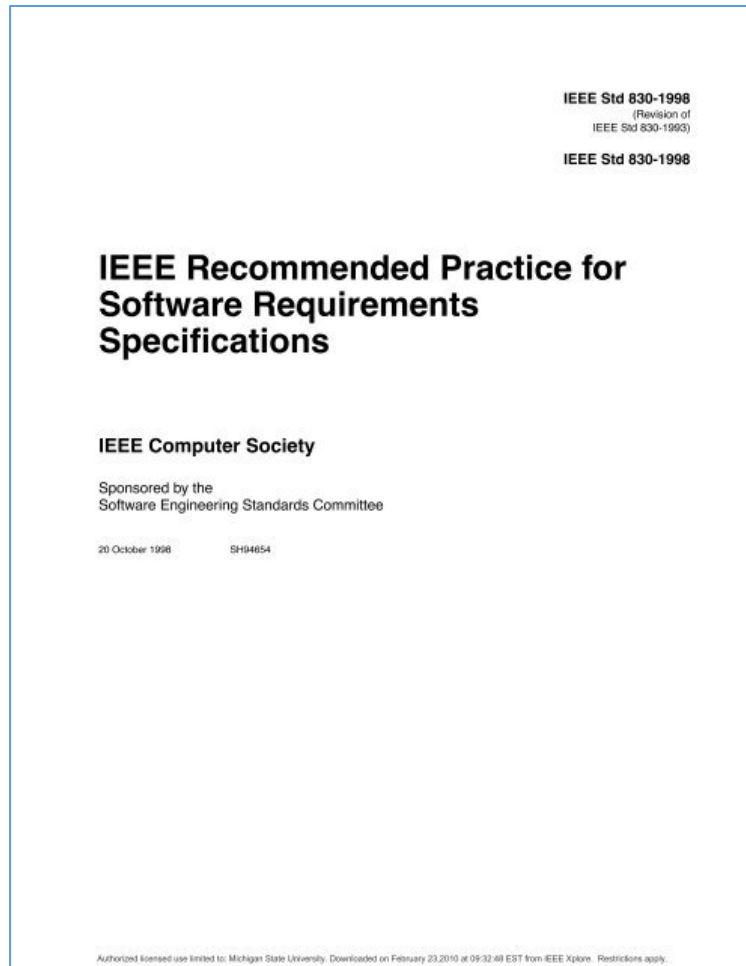


**Requirements validation**

- Consists of detailing, structuring and documenting the characteristics of the new software
- Resulting artifact: ***software requirements specification*** is a document that comprehensively describes the intended purpose, expected behavior, and functionalities of a software system to be developed
  - Purpose
  - Functional requirements
  - Non-functional requirements
  - Constraints
  - Models
  - Acceptance criteria
  - External interfaces
  - ...



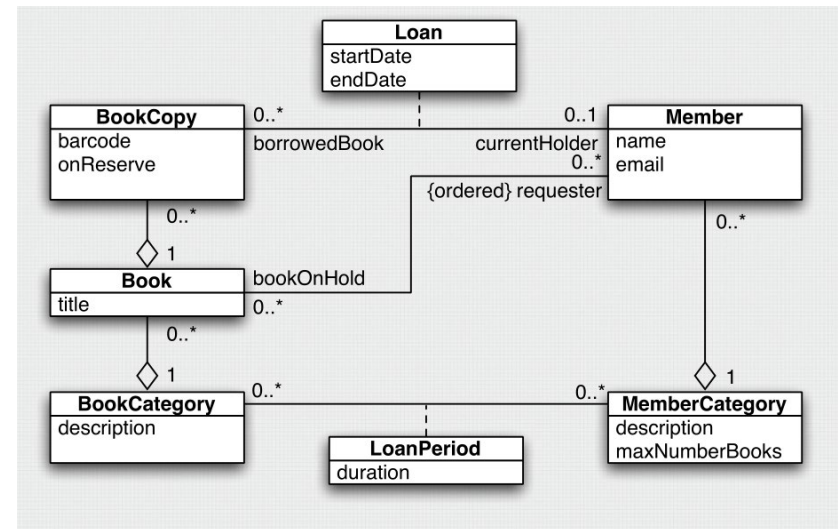
- An SRS standard (template): IEEE Std 830



- Many different notations for modeling and documenting and SRS

## 3.1.1 Sell Configured to Ordered Products.

- 3.1.1.1 The system shall display all the products that can be configured.
- 3.1.1.2 The system shall allow user to select the product to configure.
- 3.1.1.3 The system shall display all the available components of the product to configure
- 3.1.1.4 The system shall enable user to add one or more component to the configuration.
- 3.1.1.5 The system shall notify the user about any conflict in the current configuration.



*LibSystem*

*members* :  $\mathbb{P} \text{ PERSON}$

*shelved* :  $\mathbb{P} \text{ BOOK}$

*checked* :  $\text{BOOK} \rightarrow \text{PERSON}$

$\text{shelved} \cap \text{dom checked} = \emptyset$

$\text{ran checked} \subseteq \text{members}$

$\forall \text{ mem} : \text{PERSON} \bullet \#(\text{checked} \triangleright \{\text{mem}\}) \leq \text{MaxLoan}$

# Examples of modeling requirements

	Prosa Requirement	Requirement Sentence Template	In FRL $R = \left( \begin{smallmatrix} \text{then} \\ A \rightarrow B \rightarrow C \\ \text{first} \end{smallmatrix} \right)$	Graphical
R <sub>1</sub>	If button B4 has not been down during the last 5 seconds, then lamp L3 must be lit immediately after button B4 changed to down.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is not down</span> P <sub>1</sub> held the last <span style="border: 1px solid black; padding: 2px;">5 seconds</span> T <sub>1</sub> and now <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>2</sub> occurs, then must <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is lit</span> P <sub>3</sub> follow immediately.	$A = [P_1]_{t_1}^{t_2} \wedge [P_2]_{t_3}^4 \wedge (t_2   t_3) \wedge (t_1 = t_2 - T_1)$ $B = [P_3]_{t_5}^{t_6} \wedge (t_3   t_5)$	
R <sub>2</sub>	Immediately after button B4 is released while the system is in mode m1 and lamp L3 is not lit, the lamp L3 must be lit until button B4 is down again or the system leaves m1.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>1</sub> occurs, during <span style="border: 1px solid black; padding: 2px;">System is in mode m1 AND</span> <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is not lit</span> P <sub>2</sub> holds, then <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is lit</span> P <sub>3</sub> must hold immediately, until <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is up</span> OR <span style="border: 1px solid black; padding: 2px;">System is not in mode m1</span> P <sub>4</sub> hold.	$A = [[P_1]_{t_1}^{t_2} \wedge [P_2]_{t_3}^{t_4} \wedge (t_3 < t_1 < t_4)$ $B = [P_3]_{t_5}^{t_6} \wedge (t_1   t_5)$ $C = [P_4]_{t_7}^{t_8} \wedge (t_6   t_7)$	
R <sub>3</sub>	Immediately after button B4 is pressed, Lamp L3 must be red for 10 seconds.	If <span style="border: 1px solid black; padding: 2px;">Button B<sub>4</sub> is down</span> P <sub>1</sub> occurs, then <span style="border: 1px solid black; padding: 2px;">Lamp L<sub>3</sub> is red</span> P <sub>2</sub> hold immediately, until <span style="border: 1px solid black; padding: 2px;">10 seconds</span> T <sub>1</sub> elapsed.	$A = [[P_1]_{t_1}^{t_2}$ $B = [P_2]_{t_3}^{t_4} \wedge (t_2   t_3)$ $C = (t_4 = t_3 + 10s)$	

While the aircraft is in-flight and the engine is running, the control system shall maintain engine fuel flow above 1.5 lbs/sec.

Where the control system includes an overspeed protection function, the control system shall test the availability of the overspeed protection function before aircraft dispatch.

If the computed airspeed is unavailable, then the control system shall use modeled airspeed.

While the aircraft is on the ground, when reverse thrust is commanded, the control system shall enable deployment of the thrust reverser.

<https://www.jamasoftware.com/>

1. The user shall be able to search either all of the initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in the document store.
3. Every order shall be allocated a unique identifier (ORDER\_ID) which the user shall be able to copy to the account's permanent storage area.

**4.A.5** The database shall support the generation and control of configuration objects; that is, objects which are themselves groupings of other objects in the database. The configuration control facilities shall allow access to the objects in a version group by the use of an incomplete name.

**4.C.8** It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Java character set.

**9.3.2** The system development process and deliverable documents shall conform to the process and deliverables defined in DoD-Spec-####.

**7.6.5** The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

- **Unambiguous:** Requirements should be expressed in a clear and straightforward manner to avoid any confusion or misinterpretation.
- **Complete:** Requirements should cover all aspects of the intended system. They should address functional and non-functional aspects, including performance, security, and usability. Incomplete requirements may lead to gaps in the final product.
- **Consistent:** Requirements should be consistent with each other and with any existing system specifications. Inconsistencies can cause confusion and may result in a system that does not behave as intended.



- **Feasible:** Requirements should be technically and economically feasible. They should not demand the use of technologies that do not exist or are not practical within the given constraints.
- **Modifiable:** Requirements should be adaptable to changes in the project environment or evolving stakeholder needs. A system's requirements may change over time, and the documentation should be flexible enough to accommodate the changes.

# A simple case study

- Assume a medical problem domain
  - An example: a hospital
  - Entities: patient, doctor, nurse, ...



- **Ambiguous requirement:** "The system should track patient progress effectively."
  - Subjectivity (effective is open to interpretation)
  - Lack of detail (what kind of progress to be tracked)
  - Missing functionality (how progress is tracked, manual, charts, ...)
- **Improved requirement:** "The system shall automatically capture and display vital signs, medication administration records, and rehabilitation progress reports for each patient in real-time, allowing authorized users to visualize trends, compare against treatment plans, and identify potential issues early on."

- **Ambiguous requirement:** "System should be user-friendly."
- **Improved requirement:** The system should provide a graphical user interface (GUI) with intuitive navigation, consistent layout, and contextual help features. Users should be able to perform common tasks, such as admitting a patient or updating medical records, without the need for extensive training. Usability testing should be conducted to ensure that the average user can complete these tasks within a specified time frame, and any identified issues should be addressed in subsequent iterations of the software."

- **Incomplete requirement:** "The system should generate alerts for critical patients."
  - Missing definition of "critical": What define a patient as critical?
  - Unclear alert specifics: What triggers the alert? What information does it contain? Who receives it? How is it delivered (SMS, pager)?
  - No prioritization or escalation: Are there different alert levels for varying severities?
- **Improved requirement:** "The system shall continuously monitor vital signs, medications, and laboratory results for all patients admitted to the ICU or Emergency Department. When predefined thresholds are breached, or specific medical conditions suggest criticality, the system shall trigger high-priority alerts for assigned care teams. These alerts will be displayed as pop-up notifications on designated workstations and sent via SMS to on-call physicians. Each alert will provide the patient's name, specific parameter exceeding the threshold (or brief description of critical condition), timestamp, and recommended initial actions. If an alert remains unacknowledged for 5 minutes, the system shall escalate it to the attending physician's mobile phone."

- **Incomplete requirement:** "The system should manage patient information."
- **Improved requirement:** "The system should be able to capture, store, and retrieve comprehensive patient information, including personal details (name, address, contact information), medical history, current medications, allergies, and insurance details. The patient information should be easily searchable and sortable based on various criteria such as name, medical record number, or admission date. Additionally, the system should maintain an audit trail of all changes made to patient records, including the user responsible for each modification, date, and time. The patient information should be securely stored, adhering to applicable data protection regulations, and accessible only to authorized healthcare personnel with the appropriate permissions."

- **Inconsistent requirement:** "The system must be accessible to all users at all times, with a maximum login failure count of 3 attempts before an account lockout."
  - Accessibility vs. lockout: Demanding constant accessibility while enforcing a lockout contradicts itself.
- **Improved requirement:**
  - A. "The system shall offer 24/7 availability with a login retry mechanism based on a risk-adaptive approach. After exceeding 3 consecutive login attempts, the system shall progressively increase the wait time before subsequent attempts and implement additional authentication steps (e.g., CAPTCHA) for high-risk scenarios. Account lockout may be triggered only after repeated failed attempts exceeding a predefined threshold, considering specific criteria such as user-based risk profiles and suspicious activity detection."
  - B. "For scenarios requiring enhanced security, such as access to sensitive patient data, the system shall implement multi-factor authentication (MFA) in addition to the standard password login. MFA may be enforced dynamically based on user roles, access privileges, or system context."

- **Inconsistent requirement:** "The system should allow doctors to view patient records in real-time."
- **Improved requirement:** "The system should provide real-time access to patient records for authorized medical personnel, including doctors, nurses, and administrative staff. Access permissions should be role-based, ensuring that users have appropriate levels of access to patient information based on their responsibilities. Real-time access means that any updates or modifications to patient records should be immediately reflected across the system for all authorized users. The system should maintain data consistency and coherence, preventing conflicting changes by different users at the same time. Any conflicts should be resolved through a version control mechanism or a notification system to alert users about potential discrepancies in the data."



- **Infeasible requirement:** "The system must predict a patient's future health outcomes with 100% accuracy."
  - Complexity of the human body: Biological systems are inherently complex and influenced by multiple factors beyond perfect human comprehension, making perfect prediction highly unlikely.
  - Ethical considerations: Guaranteeing 100% accuracy could lead to over-reliance on the system, potentially neglecting crucial human judgment and personalized care.
  - Technological limitations: Current medical technology and data analysis methods lack the capability for flawless prediction in healthcare.
- **Improved requirement:** "The system shall leverage advanced analytics and machine learning algorithms to analyze patient data (medical history, lab results, medication usage, etc.) and generate probabilistic models for potential future health outcomes. These models will provide physicians with insights into the likelihood of various scenarios, aiding in informed decision-making and personalized care planning."

- **Infeasible requirement:** "The system should provide 100% uptime for all users."
- **Improved requirement:** "The system should aim to achieve high availability, with a target uptime of 99.99%. This target takes into account planned maintenance windows, unforeseen system issues, and any necessary updates. To enhance reliability, the system should include failover mechanisms, regular backups, and a robust disaster recovery plan. The achievement of this target will be regularly monitored and reported, and any unplanned downtime exceeding acceptable thresholds will trigger immediate investigation and resolution efforts."

- **Modifiable requirement:** "The system shall generate reports on patient demographics, diagnoses, and treatments in PDF format."
  - Limited format: Restricting reports to PDF excludes other potentially useful formats like CSV, Excel, or interactive dashboards.
  - Static content: Defining specific data points beforehand limits the report's ability to adapt to user needs and changing data requirements.
- **Improved requirement:** The system shall provide a customizable reporting system for healthcare professionals. Users shall be able to:
  - Select from a variety of report formats, including PDF, CSV, Excel, and interactive dashboards.
  - Choose which data fields to include in the report, allowing for filtering and personalization.
  - Define custom timeframes and aggregation methods for data analysis.
  - Schedule reports for automatic generation and delivery via email or other preferred channels.
  - Export reports in different formats for further analysis or integration with other tools."

- **Good structuring:** A well-structured set of requirements resembles a neatly organized library. It categorizes and prioritizes different needs, making them easier to understand and navigate
  - **Functional requirements:** Define what the system must do (e.g., scheduling appointments, generating reports).
  - **Non-functional requirements:** Outline performance, security, and usability aspects (e.g., system response time, data encryption, accessibility).
  - **User roles:** Group requirements based on who uses them (e.g., patient-facing features, doctor-specific functionalities).
  - **Priority levels:** Identify essential, desirable, and future-oriented requirements (e.g., core features vs. advanced functionalities).

- **Traceability:** Imagine requirements as individual puzzle pieces. Traceability ensures each piece connects seamlessly to the bigger picture, showing how requirements relate to each other and to different stages of development
- Requirement: "The system shall allow patients to schedule appointments online."
  - **Traceability to source:** Traceability to the source of a requirement is the ability to clearly understand and follow the origin, evolution, and justification for that requirement throughout the software development lifecycle
  - **Traceability to design document:** "The user interface will include a calendar and booking form for online appointment scheduling."
  - **Traceability to test case:** "Test case X verifies that online appointment scheduling correctly captures patients' preferred date and time slot."



**Domain engineering**



**Requirements elicitation**



**Requirements modeling  
and documentation**



**Requirements 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



- An example requirement: The system shall allow users to register on the platform using their email address and a password
- **Verification:** focuses on ensuring the completeness, consistency, and feasibility of the requirement itself, within the context of the system and other requirements
  - **Is the requirement clearly stated and well-defined?** (e.g., "Is 'platform' ambiguous? Should it specify type of platform?")
  - **Does it conflict with any other requirements?** (e.g., "Are there other login methods defined?")
  - **Is it technically feasible to implement with existing resources and technology?** (e.g., "Do we have secure password storage capabilities?")
  - **Are the UML diagrams syntactically correct?**



- An example requirement: The system shall allow users to register on the platform using their email address and a password
- **Validation:** focuses on whether the requirement actually meets the needs of the stakeholders and solves the intended problem
  - **Does this registration method meet the user's preferred way of signing up?** (e.g., "Would users prefer social media login?")
  - **Does it achieve the business goal of attracting new users to the platform?** (e.g., "Are there easier or faster registration methods?")
  - **Are there any security concerns with using email and password for authentication?** (e.g., "Is two-factor authentication necessary?")

- Requirements engineering establishes a strong foundation for software development, defining what the software should do, who it serves, and why it exists
  - This saves time, resources, and ultimately protects software projects from costly rework and failures
- By thoroughly analyzing and refining requirements before coding, we avoid building features that miss the mark or contradict user needs
  - This saves time, resources, and ultimately protects software projects from costly rework and failures.
- When requirements accurately reflect user needs and stakeholder goals, the resulting software is more likely to be intuitive, useful, and meet its intended purpose
  - This translates to greater user satisfaction, adoption, and ultimately, a successful software solution.