Human Computer Interaction
CE 382

Course Instructor: Vincent M. Nofong, Ph.D.

June 17, 2025

Introduction

Outline

- Who I am
- Course Information and Outline of CE 382
- Expected Learning Outcomes
- Rules
- Chapter Two: Establishing Requirements

Introduction

About me

- Name: **Vincent M. Nofong, PhD**
- Email: **vnofong@umat.edu.gh**
- Personal Website: https://vincentnofong.com/
- Uni website: https://www.umat.edu.gh/staffinfo/staffDetailed.php?contactID=385
- Office hours (Working days): **09:00 am - 16:00 pm GMT**
- Research interest: **data mining, trend prediction, classification, bioinformatics, artificial intelligence, machine learning**

Introduction

Course Information (CE 382)

- Credit hours: **3**
- Attendance: **10%**
- Continuous Assessment: **30%**
  - Quizzes - two or three
  - Group assignment - one (application development)
  - Group presentations
- End of Semester: **60%**

Introduction

Course Outline (CE 382)

1. Interaction Design
2. Establishing Requirements
3. Prototyping
4. Data Gathering and Analysis
5. Cognitive Aspects of Design
6. Social and Emotional Interactions
7. User Interfaces
8. Evaluations

Introduction

Expected Learning Outcomes (CE 382)

Students should understand and be able to:

1. Explain the characteristics of good and bad interaction design and use them to evaluate HCIs
2. Explain the characteristics of users that influence HCI and use them to inform user interface development
3. Explain, analyze and develop interaction evaluations
4. Explain and develop requirements for interaction design
5. Construct interactions using evaluation-based iterative process for directing the design of user interfaces.

Introduction

Reference Materials

1. Preece, J., Rogers, Y. and Sharp, H. (2023), Interaction Design: Beyond Human-Computer Interaction, John Wiley & Sons Ltd, Hoboken, U.S.A., 6th Edition, 716 pp. - slides are based on this reference
2. Lazar, J., Feng, J. H. and Hochheiser, H. (2017), Research Methods in Human-Computer Interaction, Morgan Kaufmann, Burlington, U.S.A., 2nd Edition, 560 pp.
3. Shneiderman B., Plaisant C., Cohen M. and Jacobs, S. (2016), Designing the User Interface, Pearson Publishers, 6th Edition, 616 pp.

Introduction

Rules

1. Feel free to ask questions in class, unless they are too "personal".
2. Students should not be late for lectures or practicals.
3. Students should attend all lectures and practicals.
4. **In case you are unable to attend lectures or will be late, send me an email - at least 30 minutes before lectures.**
5. Students should do and submit all assignments before the given deadline.
6. Unless otherwise permitted, students should not use their mobile phones in class - note usage of Laptops/Desktops is permitted.

HCI CE 382
Chapter Two: Establishing Requirements

Course Instructor: Vincent M. Nofong, Ph.D.

June 17, 2025

Establishing Requirements

Group Assignment

- Group A should divide themselves into 4 groups
- Group B should divide themselves into 4 groups
- I want the list of members on each group by our next meeting, the application they are developing, and the user category.

Establishing Requirements

Group Assignment: Tasks

Question 1: Design and implement a Health and Wellness App:

1. For young children and teenagers - 18 years and below
2. For elderly people - 50 years and above

Question 2: Design and implement an Educational Platform App:

1. For young children and teenagers - 18 years and below
2. For elderly people - 50 years and above

Establishing Requirements

Group Assignment: Tasks

Question 3 - Design and implement a Digital Library App:

1. For young children and teenagers - 18 years and below
2. For elderly people - 50 years and above

Question 4: Design and implement an Entertainment Hub App:

1. For young children and teenagers - 18 years and below
2. For elderly people - 50 years and above

One group per user category.
First Come, First Served, aka, fight among yourselves to decide which question your group picks

Establishing Requirements

What are Requirements?

- Requirements are statements that define what a product is expected to do or how it will perform.
- They come in different forms and levels of abstraction.
- User stories are commonly used in agile development contexts to capture requirements.
- User stories follow a specific format:
  - *"As a <role>, I want <behavior> so that <benefit>."*

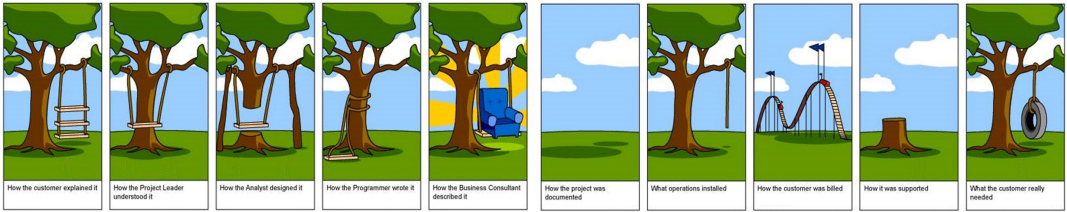
Establishing Requirements

What are Requirements?

- Example User Stories for a Travel Organizer:
  - As a <traveler>, I want to <save my favorite airline for all my flights> so that <I will be able to collect air miles>.
  - As a <travel agent>, I want <my special discount rates to be displayed to me> so that <I can offer my clients competitive rates>.
- These user stories provide specific examples of how requirements can be expressed for a travel organizer, focusing on the needs and desired benefits of different user roles.

Establishing Requirements

Why are Requirements Important?



- Miscommunication is most commonly observed during the requirements activity stage.
- Ensuring accurate and clear requirements is of utmost importance.

Establishing Requirements

Functional vs. Non-functional Requirements

Functional Requirements:

- Define the specific behaviours and functionalities that the system should exhibit.
- They focus on the specific actions and outputs of the system.

Non-functional Requirements:

- Elaborate on the performance characteristics or constraints that the system should adhere to.
- They provide criteria for evaluating aspects such as performance, security, usability, and reliability.

Establishing Requirements

Functional vs. Non-functional Requirements

Functional Requirements:

- Example: The system must send an email whenever a certain condition is met, such as an order being placed or a customer signing up.

Non-functional Requirements:

- Example: Emails should be sent with a latency of no greater than 12 hours from the corresponding activity.

Establishing Requirements

System Stakeholders

- System stakeholders are individuals or organizations who are impacted by the system and have a legitimate interest in its development, implementation, or usage.
- **Stakeholder Types**:
  - End Users - directly interact with the system, utilizing its functionalities to perform tasks or achieve specific goals
  - System Managers - responsible for overseeing the operation, maintenance, and administration of the system
  - System Owners - have ownership or financial responsibility for the system
  - External Stakeholders - entities external to the system who are influenced by or have an interest in the system (customers, regulatory bodies, partners or general public)

Establishing Requirements

Problems of Requirements Elicitation (1/2)

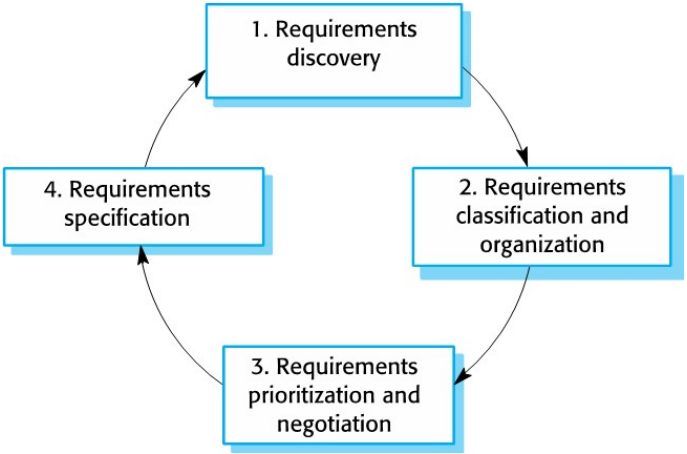
- Stakeholders' Unclear Understanding
  - Stakeholders may have difficulty articulating their needs and desires accurately, leading to a lack of clarity in requirements.
- Stakeholders' Terminology
  - Stakeholders may express their requirements using their own jargon or language, making it challenging to translate their needs into actionable system requirements.
- Conflicting Requirements
  - Different stakeholders may have diverse and sometimes conflicting requirements, creating challenges in prioritizing and reconciling these conflicting demands.

Problems of Requirements Elicitation (2/2)

- Organizational and Political Influences
  - Organizational and political factors can influence the system requirements, leading to biases and preferences that may not align with the optimal solution.
- Changing Requirements
  - Requirements may evolve and change throughout the analysis process.
  - New stakeholders may emerge, and the business environment may undergo transformations, necessitating adjustments to the initial set of requirements.

Establishing Requirements

The Requirements Elicitation and Analysis Process



The Requirements Elicitation and Analysis Process

Process Activities (1/2)

- Step 1 - Requirements Discovery:
  - Engaging with stakeholders to uncover their requirements, including both functional and domain-specific requirements.
  - This stage involves active communication and information gathering.
- Step 2 - Requirements Classification and Organization:
  - Grouping and organizing related requirements into coherent clusters to facilitate a structured and systematic understanding of the overall system needs.
  - This step aids in identifying dependencies and relationships between different requirements.

The Requirements Elicitation and Analysis Process Process Activities (2/2)

- Step 3 - Prioritization and Negotiation:
  - Assigning priority levels to requirements based on their importance and impact on the system's success.
  - Resolving conflicts and discrepancies between different stakeholder requirements through negotiation and consensus-building.
- Step 4 - Requirements Specification:
  - Documenting the identified requirements in a clear, concise, and unambiguous manner.
  - These specifications serve as input for the subsequent iterations of the development process, ensuring that the requirements are well-documented for further analysis and implementation.

Establishing Requirements

Establishing Requirements

- The process of establishing requirements involves determining what users want and need for a successful system implementation.
  - What do users want? What do users 'need'?
- Requirements often require clarification, refinement, completion, and sometimes even re-scoping to ensure they accurately represent user expectations.
- Input: Requirement document or other relevant sources of information
- Output: Stable requirements that effectively capture user needs and system expectations

Establishing Requirements

Why "Establish" Requirements?

- Establishing requirements is crucial because they arise from a deep understanding of user needs, ensuring that the system aligns with their goals and objectives.
- By establishing requirements, we can justify their relevance and establish clear relationships with relevant data and user insights.
- This process enables us to gather comprehensive and accurate requirements that serve as a foundation for designing and developing a successful system.

Establishing Requirements

Categories of Requirements (1/5)

- Functional Requirements:
  - Statements that define the services the system should provide, its expected reactions to specific inputs, and desired behavior in various situations.
  - May also include statements about what the system should not do.
- Non-Functional Requirements;:
  - Constraints placed on the system's services or functions, such as timing limitations, development process requirements, and adherence to standards.
  - Typically apply to the system as a whole, rather than individual features or services.

Establishing Requirements

Categories of Requirements (2/5)

- Environment or Context of Use
  - **Physical Environment**: consider factors such as dust, noise, vibration, lighting, heat, humidity, etc. E.g., in a hospital setting, these factors may impact the design and functionality of the system.
  - **Social Environment**: Address requirements related to collaboration, coordination, data sharing, distributed systems, synchronous or asynchronous communication, and privacy considerations.
  - **Support**: Specify requirements for user support, communications structure, infrastructure, and availability of training to ensure smooth operation and user satisfaction.
  - **Technical Environment**: Identify the technologies on which the system will run or with which it needs to be compatible, ensuring compatibility and interoperability with existing systems or platforms.

Categories of Requirements (3/5)

- User Characteristics:
  - Consider the characteristics of the users who will interact with the system.
  - Factors to consider include educational background, personal circumstances, abilities, and skills.
- User Profiles:
  - Develop user profiles that capture relevant information about the users, including their roles, preferences, and specific needs.
  - User profiles help guide the design and customization of the system to better align with the diverse user characteristics and requirements.

Establishing Requirements

Categories of Requirements (4/5)

- System Use:
  - Identify the different user profiles based on their experience and frequency of system use.
  - **Novice Users**: These users may require prompts, constraints, and clear instructions to navigate the system effectively.
  - **Expert Users**: These users may value flexibility and desire greater access and control over the system.
  - **Frequent Users**: These users may benefit from shortcuts or efficient ways to accomplish tasks quickly.
  - **Casual/Infrequent Users**: These users may require clear menu paths and intuitive navigation to facilitate ease of use.

Categories of Requirements (5/5)

- Domain Requirements:
  - Constraints imposed on the system due to the specific domain of operation it belongs to.

Establishing Requirements

Data Gathering Techniques for Requirements

- Interviews:
  - Involve direct conversations with stakeholders to gather requirements.
  - Props such as sample scenarios or prototypes can be utilized to facilitate discussions and enhance understanding.
  - Interviews are effective for exploring various issues and allowing development team members to establish a connection with stakeholders.
- Focus Groups:
  - Entail group interviews with multiple stakeholders simultaneously.
  - They are beneficial for obtaining a consensus view on requirements or identifying areas of conflict.
  - However, it is important to be cautious of dominant individuals who may influence the group dynamics.

Establishing Requirements

Data Gathering Techniques for Requirements

- Questionnaires:
  - Questionnaires are frequently used alongside other data gathering techniques.
  - They provide the opportunity to collect both quantitative and qualitative data.
  - Questionnaires are particularly useful when seeking specific responses from a large and geographically dispersed group of people.
- Researching Similar Products:
  - Conducting research on similar products in the market can help prompt and inspire requirements.
  - By analyzing existing products, designers can gain insights into industry trends, best practices, and potential functionalities.

Establishing Requirements

Data Gathering Techniques for Requirements

- Direct Observation:
  - Direct observation involves actively observing stakeholders while they perform their tasks.
  - It provides valuable insights into the nature and context of the tasks being performed.
  - Direct observation is particularly useful for gaining a deep understanding of stakeholders' behaviors and needs.
- Indirect Observation:
  - Indirect observation, such as task logging, is not commonly used in the requirements gathering process.
  - It involves recording and documenting stakeholders' current tasks and activities.
  - Indirect observation can be beneficial for capturing a comprehensive view of stakeholders' tasks and workflows.

Establishing Requirements

Data Gathering Techniques for Requirements

- Studying Documentation:
  - Studying documentation involves reviewing manuals, procedures, and rules.
  - It provides valuable data on the steps involved in specific activities and any regulations or guidelines governing those tasks.
  - Documentation is a good source for understanding legislation and obtaining background information.
  - Documentation should not be used in isolation but in conjunction with other data gathering techniques.

Establishing Requirements

Bringing Requirements to Life

- Scenarios and Personas:
  - These are informal narrative stories that depict specific situations or interactions.
  - They are simple, natural, and personalized, focusing on individual experiences rather than generalizations.
  - Scenarios help in understanding the context, goals, and challenges of users' interactions with a system.
- Use Cases:
  - They assume direct interaction with a system and require a detailed understanding of the interaction.
  - They outline specific actions, inputs, and expected outcomes of a user's interaction with the system.
  - Use cases are beneficial for capturing system behavior and identifying key functionalities.

Bringing Requirements to Life

- Essential Use Cases:
  - Essential use cases provide a higher-level abstraction, focusing on the core functionalities of the system.
  - They abstract away from implementation details and specific interactions.
  - Essential use cases help in capturing the essential requirements and system capabilities without making assumptions about the specific details of the interaction.

Establishing Requirements

Bringing Requirements to Life

Scenarios and Personas - Relationship

Establishing Requirements

Users' Capabilities and Variations

Humans vary in many dimensions:

- size of hands may affect the size and positioning of input buttons
- motor abilities may affect the suitability of certain input and output devices
- height if designing a physical kiosk
- strength - a child's toy requires little strength to operate, but greater strength to change batteries
- disabilities (e.g. sight, hearing, dexterity)

The diverse capabilities and variations among users should be considered in establishing requirements

Establishing Requirements

Creating Personas: Bringing Users to Life

- Personas capture a set of user characteristics, representing fictional but synthesized profiles based on real users.
- They provide a human-centered approach to design by bringing users to life with names, characteristics, goals, and personal backgrounds.
- Developing a small set of personas helps designers gain a deeper understanding of user needs and preferences.

By leveraging personas, designers can better empathize with users, align design decisions with user needs, and create more user-centered and engaging experiences.

Creating Personas: Example Persona

Lena, 50

Lena works in London as a civil servant. She lives with her partner in a commuter town, and they both own a car.

She commutes to London by train, taking an early morning service which takes over an hour. She leaves the house early and often doesn't get home until after 7pm. She drives 15 minutes to the station and usually arrives 5 minutes before the train departs. It costs a lot to park there everyday, but there are no buses direct from her house to the station.

Lena enjoys her job but finds that she is often so busy travelling from one meeting to another that she is left with little time to complete work. At the weekends, she and her partner drive to the countryside to go walking or to visit friends.

"I wish there was a cheaper way of getting to the station – parking is so expensive and often very limited – but the car is so convenient"

"I often wonder how taxi drivers choose their routes – I feel uneasy when they don't follow their satnav"

Lena has two Android smartphones – one provided by her employer and one for personal use.

She has an Apple laptop for business use. It has to go everywhere with her as it contains confidential files.

She is always relieved if there is a charger available in the taxi so she can make sure her laptop is charged.

Travel & Transport

Top 3 modes of transport	Top 3 reasons for taxi usage
Train	Business
Shared car	Leisure
Taxi	Holidays

Willingness to share a taxi

Technology and Income

Technology acceptance

Openness to experience

Budget

ServCity

Example Scenario for Group Travel Organizer

Example 1

The Thomson family enjoy outdoor activities and want to try their hand at sailing this year. There are four family members: Sky (8 years old), Eamonn (12 years old), Claire (32), and Will (35). One evening after dinner they decide to start exploring the possibilities. They want to discuss the options together but Claire has to visit her elderly mother so will be joining the conversation from her mother's house down the road. As a starting point, Will enters an idea they had been discussing over dinner – a sailing trip for four novices in the Mediterranean. The system supports users to log on from different locations and use different devices so that all members of the family can interact easily and comfortably with it wherever they are. The system's initial suggestion is a flotilla, where several crews (with various levels of experience) sail together on separate boats. Sky and Eamonn aren't very happy at the idea of going on vacation with a group of other people, even though the Thomson's would have their own boat. The travel organizer shows them descriptions of flotillas from other children their ages and they are all very positive, so eventually, everyone agrees to explore flotilla opportunities. Will confirms this recommendation and asks for detailed options. As it's getting late, he asks for the details to be saved so everyone can consider them tomorrow. The travel organizer messages them a summary of the different options available.

Establishing Requirements

Creating Personas: Example Persona Travel Organizer (1/2)

Family traveler

"I want a travel organiser that will offer me a range of potential vacations that suit our needs"

Age: 35
Work: Plumber
Family: Married, two children

Personality

Introvert — Extrovert
Thinking — Feeling
Sensing — Intuition

Organised | Practical | Expects high standard

Goals

- To book comprehensive travel quickly
- To find a trip that meets the needs of the whole family
- To feel supported and guided from the beginning of the booking experience right to the end.

Frustrations

- Wasting time filling in forms
- Too much irrelevant information
- Existing systems tend to be too diverse and complicated

Bio

Will loves to take his family on adventure holidays to explore new challenges. His children, Sky (8) and Eamonn (15) are old enough to take part in several sporting activities and he wants to make the most of this before they no longer want to go on trips with him and his wife, Claire. He likes the fact that choosing travel options is so much easier than it used to be, but is frustrated by the many different sources and disjointed options that this can result in. He wants a travel organiser that can provide clear support for family holidays while offering as wide a choice as possible.

Motivation

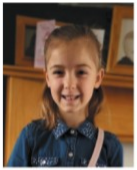
Price
Comfort
Choice

Favourite destinations

Establishing Requirements

Creating Personas: Example Persona Travel Organizer(2/2)

Young traveler



"I want a travel organiser that will allow all of us to choose the vacation together"

Age: 8
Work: Schoolgirl
Family: Mum Dad and Eamonn (15)

Personality

Introvert — Extrovert
Thinking — Feeling
Sensing — Intuition

Energetic | Inquisitive | Likes reading

Goals

- To find a good vacation without any fuss
- To find a destination with other children her age
- To make sure that the travel time is short

Frustrations

- Sitting around discussing things for too long
- Not getting clear answers to her questions
- Feeling that everything is organised for adults and not children her age

Bio

Sky likes having adventures. She is very energetic and takes part in lots of sporting activities at school, such as gymnastics and swimming. She enjoys playing games with her older brother, Eamonn. Sky is keen to make new friends, but is also happy sitting reading a book, painting or making a model. She likes going to visit new places but expects to see something familiar, such as playground or food that she recognises!
The most important thing for her is that she can go on vacation with her family where there will be something for everyone to do - but especially for her and Eamonn.

Motivation

Fun
Comfort
Choice

Favourite destinations

Establishing Requirements

Use Cases

- Use cases are a valuable tool for capturing functional requirements and depicting interactions with a system.
- They can be used in the design process or as a means to capture requirements.
- Use cases provide step-by-step descriptions of interactions between users and the system.
- They outline the specific actions, inputs, and expected outcomes of each interaction.

Establishing Requirements

Use Cases Styles

1. Essential Use Cases: These focus on the division of tasks without delving into implementation details.
2. Use Case with Normal and Alternative Courses: This style provides more detailed descriptions, including alternative paths and exceptional scenarios.

Establishing Requirements

Benefits of Use Cases

- Use cases facilitate a clear understanding of the system's functionality and how it interacts with users.
- They help in capturing requirements in a structured and comprehensive manner.
- Use cases can be used as a foundation for design decisions and as a reference throughout the development process.

Establishing Requirements

Case Study: Use Case for Travel Organizer (1/2)

1. The system displays options for investigating visa and vaccination requirements.
2. The user chooses the option to find out about visa requirements.
3. The system prompts user for the name of the destination country.
4. The user enters the country's name.
5. The system checks that the country is valid.
6. The system prompts the user for her nationality.
7. The user enters her nationality.

Establishing Requirements

Case Study: Use Case for Travel Organizer (2/2)

8. The system checks the visa requirements of the entered country for a passport holder of her nationality.
9. The system displays the visa requirements.
10. The system displays the option to print out the visa requirements.
11. The user chooses to print the requirements.

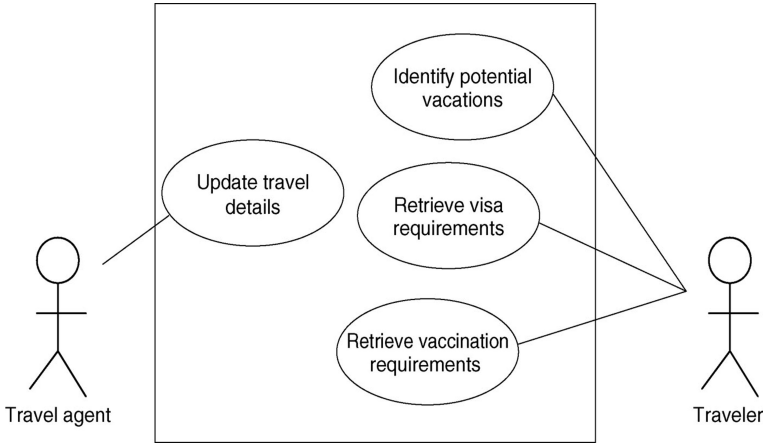
Establishing Requirements

Case Study: Alternate Courses for Travel Organizer

6. If the country name is invalid:
    1. The system displays an error message.
    2. The system returns to step 3.
8. If the nationality is invalid:
    1. The system displays an error message.
    2. The system returns to step 6.
9. If no information about visa requirements is found:
    1. The system displays a suitable message.
    2. The system returns to step 1.

Establishing Requirements

Case Study: Use Case for Travel Organizer



Is the Use Case complete? What are the issues?

Establishing Requirements

Case Study: Essential (Business) Use Case for Travel Organizer

retrieve Visa

USER INTENTION	SYSTEM RESPONSIBILITY
find visa requirements	request destination and nationality
supply required information	obtain appropriate visa info
obtain copy of visa info	offer info in different formats
choose suitable format	provide info in chosen format

Establishing Requirements

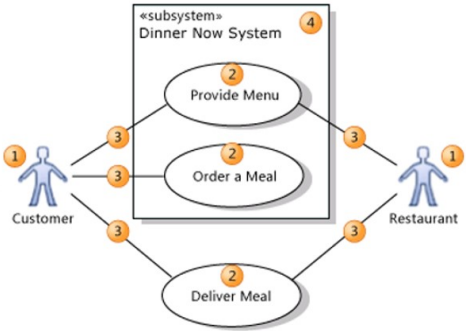
Use Case Modeling

- Use cases were initially developed to facilitate requirements elicitation and are now an integral part of the Unified Modeling Language (UML).
- Each use case represents a distinct task that involves external interactions with a system.
- *Actors* in a use case can be individuals or other systems interacting with the system under consideration.
- Use cases are depicted diagrammatically to provide an overview of the interactions, and they are also described in more detailed textual form.

Establishing Requirements

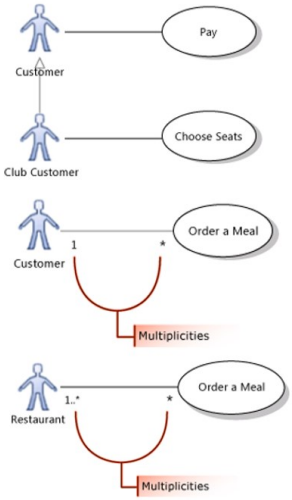
Use Case Modeling Basics (1/5)

- An *actor* (1) is
  - a class of person, organization, device, or external software component that interacts with your system. Example actors are **Customer**, **Restaurant**, **Temperature Sensor**, **Credit Card Authorizer.**
- A *use case* (2)
  - represents the actions that are performed by one or more actors in the pursuit of a particular goal. Example use cases are **Order Meal**, **Update Menu**, **Process Payment**.
- On a use case diagram, use cases are associated (3) with the actors that perform them.
- Your *system (4)* is
  - whatever you are developing. It might be a small software component, whose actors are just other software components; or it might be a complete application; or it might be a large distributed suite of applications deployed over many computers and devices. Example subsystems are **Meal Ordering Website**, **Meal Delivery Business**, **Website Version 2**.
- A use case diagram can show which use cases are supported by your system or its subsystems.

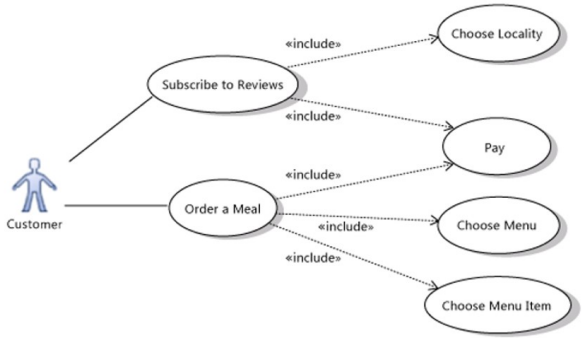


Use Case Modeling Basics (2/5)

- You can draw a **Generalization** link between Actors.
  - The specialized actor, such as Club Customer in the example, inherits the use cases of the generalized actor, such as Customer.
- The association between an actor and a use case can show a *multiplicity* at each end.
  - **1** to state that exactly one instance of this role participates in each link.
  - **1..*** to state that one or more instance of this role participate in each link.
  - **0..1** to state that participation is optional.
  - * to state that zero or more instances of this role participate in the link.
  - In the illustration, one or more restaurants can take part in fulfilling the same meal order.



Use Case Modeling Basics (3/5)

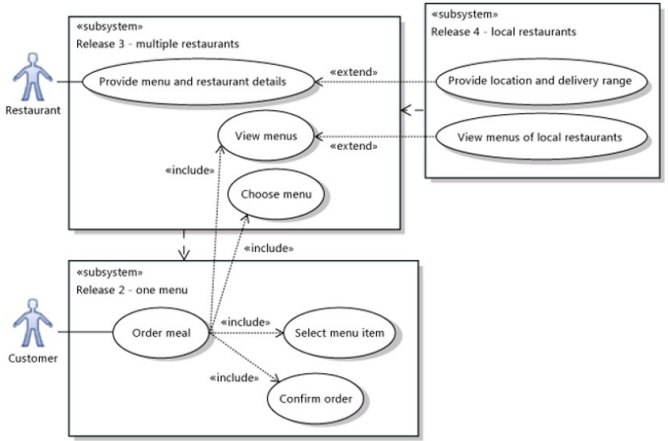


- Use an **Include** relation to show that one use case describes some of the detail of another. In the illustration, **Order a Meal** includes **Pay**, **Choose Menu**, and **Choose Menu Item**. Each of the included, more detailed use cases is a step that the actor or actors might have to perform to achieve the overall goal of the including use case. The arrow should point at the more detailed, included use case.

Use Case Modeling Basics (4/5)

- Use an Extend link to show that one use case may add functionality to another use case under certain circumstances. The arrow should point at the main, extended use case.
- For example, the **Login** use case of a typical Web site can include **Register New User** - but only when the user does not already have an account.

Use Case Modeling Basics (5/5)



- You can use different subsystem boundaries to illustrate different versions of the system.
- Use **Dependency** relations to link subsystems representing different versions or variants.

Establishing Requirements: Software Requirement

Importance of Software Requirements

- Software requirements provide the essential groundwork for software design and development, outlining the functionality and behavior expected from the system.
- A well-written set of requirements will clearly define what a software system must do in order to satisfy customer or user needs.
- Once written, software requirements serve as a way for each stakeholder to do their job effectively and stay in alignment with the other members of the project team.

Establishing Requirements: Software Requirement

Characteristics of Good Software Requirements (1/2)

- Concise: Requirements should be expressed in clear and concise language, avoiding unnecessary complexity and wordiness.
- Precise: Requirements should be specific, well-defined, and unambiguous, leaving no room for interpretation or confusion.
- Clear: Requirements should be free from ambiguity, jargon, and technical terms that may be unclear to stakeholders.
- Complete: Requirements should include all the necessary details and information required for successful implementation, leaving no crucial aspects or functionality undocumented.

Establishing Requirements: Software Requirement

Characteristics of Good Software Requirements (2/2)

- Non-contradictory: Requirements must not contradict each other or any previously established requirements to ensure consistency and coherence.
- Testable: Requirements should be formulated in a way that makes them easily testable and verifiable, allowing for effective validation and quality assurance.
- Prioritized: Requirements should be organized and prioritized based on their importance and impact, ensuring that the most critical aspects are addressed first.

Establishing Requirements: Software Requirement

How to write software requirements - The DO's

1. User-Centric Writing:
   - When writing software documentation, adopt the perspective of the user.
   - Focus on their needs, goals, and experiences throughout the document.

BAD REQUIREMENT	GOOD REQUIREMENT
The shopping cart has a list of items and the total purchase price.	As a customer I want to be able to view a list of the items in my cart along with a total purchase price, so that I can decide quickly if I want to move forward with checkout.

How to write software requirements - The DO's

2. Implementation-Neutral Requirements:
   - Formulate requirements without being tied to specific implementation technologies or platforms.
   - Focus on what the software system should achieve rather than how it should be implemented.

BAD REQUIREMENT	GOOD REQUIREMENT
The application must display the onboarding guide using a modal window with a width of 500px and a height of 300px.	The user should be able to view the onboarding guide regardless of what screen they are on.

Establishing Requirements: Software Requirement

How to write software requirements - The DO's

3. Involve stakeholders early and often:
   - Consult stakeholders at the beginning of the requirement process:
     - Get their input on project scope, intended audience, and objectives.
     - Understand their perspectives and expectations to align with project goals.
   - Seek stakeholder input on draft requirements before development:
     - Ensure their needs and preferences are considered.
     - Incorporate valuable insights to refine and enhance the requirements.
   - Notify stakeholders of changes to requirements or product scope during development:
     - Keep stakeholders informed to maintain transparency and manage expectations.
     - Address concerns or potential impact of changes to ensure stakeholder satisfaction.

Establishing Requirements: Software Requirement

How to write software requirements - The DO's

4. Analyze, Refine, and Decompose Requirements:
   - Evaluate High-Level Requirements:
     - Assess feasibility, reliability, and verifiability.
     - Resolve issues by adjusting budget, schedule, or modifying requirements.
   - Analyze the Requirements:
     - Ensure completeness, consistency, feasibility, balance, verifiability, and human factors.
   - Define Derived Software Requirements:
     - Complete definition and examine consistency, feasibility, and implementation impact.
     - Monitor size volatility of derived requirements.

How to write software requirements - The DO's

4. Analyze, Refine, and Decompose Requirements:
   - To demonstrate the process, let's break down the requirement on the left into three more specific requirements:

BAD REQUIREMENT	GOOD REQUIREMENT
As a student I need to be able to complete lessons within my eLearning courses.	1. As a student I need to be able to watch video lessons within my eLearning courses. 2. As a student I need to be able to read text-based lessons within my eLearning courses. 3. As a student I need to be able to progress through each lesson in order, without jumping ahead to future lessons within my eLearning courses.

How to write software requirements - The DO's

5. Specify the Priority of Requirements:
   - Identify Decision Makers: those to decide on requirement priorities.
   - Establish Priority Labels: a set of priorities such as "must have", "should have", and "nice to have", or a numerical scale from 1 to 10.
   - Define Criteria: that justify a requirement's priority level.

PRIORITY	DEFINITION
1	Highest priority, core functionality and/or must have basic customer success in MVP
2	Medium priority, ideally is included in the MVP, as it enables more advanced functionality
3	Lowest priority, would be nice to have, but not required for MVP

   - By establishing a prioritization matrix and involving the team in the process, priorities can be set effectively based on the software's end-goals, project budget, and timeline. (MVP = Minimum Viable Product)

Establishing Requirements: Software Requirement

How to write software requirements - The DO's

6. Systematically Track Requirement Changes:
   - Recognize the Need for Change - requirements are subject to changes (unforeseen issues, budget constraints, etc.) during the software development cycle.
   - Implement a Change Management System - establish a structured process to manage requirement changes effectively with roles and responsibilities for stakeholders involved in the change management process.
   - Capture and Document Changes - the reasons, impact, and implications of each change to maintain a comprehensive record.
   - Assess and Analyze Changes - the risks associated with implementing the changes
   - Obtain Approval and Communication - seek approval from stakeholders and communicate requirement changes to project team

Establishing Requirements: Software Requirement

How to write software requirements - The DONT's

1. Avoid Ambiguous Language and Technical Jargon:
   - Be Specific and Clear - use precise and unambiguous language to document requirements.
   - Define Key Concepts - define terms such as "easy to use" and "intuitive" to provide a shared understanding as well as specify the functionalities that contribute to these characteristics.
   - Eliminate Technical Jargon -strive for simplicity and clarity to ensure all stakeholders can comprehend the requirements.

BAD REQUIREMENT	GOOD REQUIREMENT
The system should provide an intuitive user interface that is easy to use.	The user should always be able to access the Homepage in one-click.

Establishing Requirements: Software Requirement

How to write software requirements - The DONT's

2. Avoid Excluding Acceptance Criteria:
   - Acceptance criteria are the conditions a software product must meet to be accepted by a user, a customer, or other systems.
   - Clear acceptance criteria are essential for successful software development.
   - They prevent misinterpretation and ensure alignment with stakeholders.
   - Acceptance criteria provide measurable benchmarks for quality assurance.
   - Including acceptance criteria enhances communication and mitigates misunderstandings.
   - They enable validation of requirement fulfillment and satisfaction of stakeholder expectations.

How to write software requirements - The DONT's

2. Avoid Excluding Acceptance Criteria:

BAD REQUIREMENT	GOOD REQUIREMENT
The user must be able to search for employees by department.	The user must be able to search for employees by department. 1. The user must be able to select a department from a list of available departments. 2. The search must return a list of employees in the selected department. 3. The search results must include the employee's name, department, and title. 4. The search must be completed within 5 seconds.

How to write software requirements - The DONT's

3. Avoid Neglecting to use a Standardized Format:
   - Standardized formats for software requirements enhance clarity and understanding.
   - Specification consistency improves communication among stakeholders.
   - A standardized format facilitates future maintenance and updates.
   - Using a common format reduces ambiguity and misinterpretation.

BAD REQUIREMENTS	GOOD REQUIREMENTS
1. The application must be able to track and display sales data in real-time. 2. Users must be able to access the application from any device with a web browser. 3. All pages must be secure and password protected.	Admins must be able to pull a real-time sales data report. Users must be able to access the application from any device with a web browser. Users should be required to enter a password before accessing any pages of the application.

How to write software requirements - The DONT's

4. Avoid Making Technical Assumptions:
   - Avoid making technical assumptions in software requirements.
   - Clearly specify technical requirements in nonfunctional requirements.
   - Focus on functionality from the user's perspective rather than specific technologies.
   - Improve clarity and flexibility by decoupling functional requirements from technical implementation details.

BAD REQUIREMENT	GOOD REQUIREMENT
Real-time analytics of user activity data should be stored in the relational database.	Admins should have access to real-time analytics of user activity data.

How to write software requirements - The DONT's

5 Avoid Adding Conflicting Requirements:

- Conflicting requirements can cause confusion, ambiguity, and project delays.
- Carefully review requirements to identify and resolve conflicts.
- Revise conflicting requirements to provide clarity and consistency.
- Ensure stakeholders and development teams are aware of and address conflicting requirements.

BAD REQUIREMENT	GOOD REQUIREMENT
1. Team leaders should have full access to reports. 2. Admins should be able to assign "read only" or "edit" permissions to each team leader individually.	1. Team leaders should, by default, be given "edit" access to reports. 2. Admins should be able to change the reports permissions for individual team leaders to assign them either "edit" or "read only" access.

Establishing Requirements: Software Requirement

How to write software requirements - The DONT's

6. Avoid Neglecting Non-Functional Requirements:
   - Non-functional requirements are crucial in software requirement specifications.
   - Neglecting non-functional requirements can result in systems that fail to meet stakeholder expectations.
   - Non-functional requirements define system quality, security, and reliability.
   - Include non-functional requirements in the software requirement specification document.
   - Address areas such as performance, scalability, security, usability, and maintainability.

How to write software requirements - The DONT's

6. Avoid Neglecting Non-Functional Requirements:
   - Nonfunctional requirements should address:

Security	Will you be storing PII or other sensitive data? Does your company have any specific security standards that should be adhered to?
Capacity	What kind of data storage requirements do you have? Do you expect your needs to change over time?
Compatibility	What are the minimum hardware requirements? What are the technical limitations that should be considered? Are there any specific external interface requirements?
Reliability	Do users need 24/7 access? What are the acceptable down times for your system to still meet a user's basic needs?
Scalability	What is the maximum load that should be able to be handled? Consider both data and user traffic.
Usability	Are there specific UX standards that should be followed?

Establishing Requirements: Software Requirement

Group Assignment
List of members for each group?