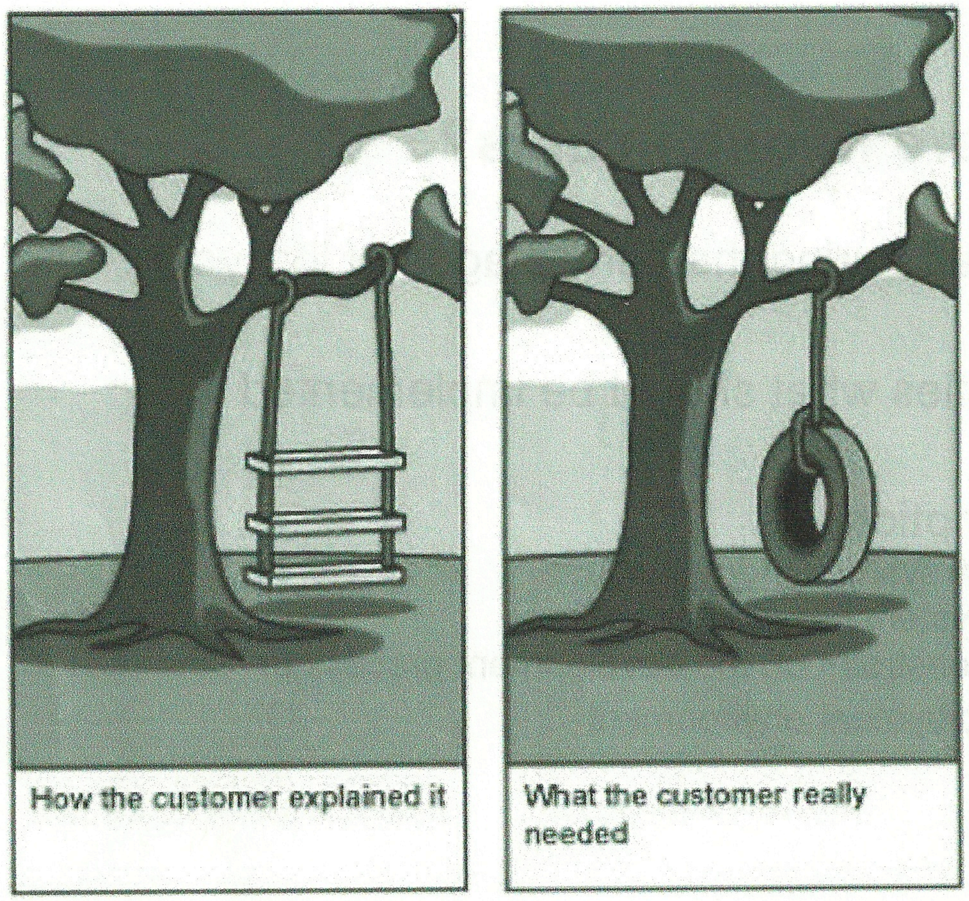
COSC 310: Software Engineering

Dr. Bowen Hui
University of British Columbia Okanagan
bowen.hui@ubc.ca


How the customer explained it
What the customer really needed

Scenario

- your client asks you to...:
  - "Write me an app that determines the most popular professor in the university."
- where do you start?
- will you be able to write a program that meets the client's expectations?

What are Requirements?

- defined during the early stages of lifecycle
- specifies what should be implemented
- descriptions of:
  - how system should behave
  - a system property or attribute
  - a constraint on the development process or deployment environment

Examples

- user-level facility
  - "The word processor must include a spell checking and correction command."
- general system property
  - "The system must ensure that personal information is never made available without authorization."
- specific constraint on system
  - "The sensor must be polled 10 times per second."
- constraint on development process
  - "The system must be developed using COBOL."

System Stakeholders

- **stakeholders** = those who will be affected by the system, or those who can influence the system requirements

- Ex: automated railway signalling system
  - operators responsible for running it
  - train crew
  - railway managers
  - passengers
  - equipment installation and maintenance engineers
  - safety certification authorities

- what about: a certified **accessible** website?

Requirements Engineering

- discipline that covers activities involving:
  - discovering requirements
  - documenting requirements
  - maintaining requirements

- "engineering" highlights the systematic and repeatable process

- ensure requirements are:
  - complete
  - consistent
  - relevant

Requirements Document

- the output of the requirements elicitation and analysis process
- official statement of system requirements

- readers:
  - client
  - end-users
  - developers

- also known as:
  - software requirements specification (SRS)
  - functional specification

Writing Requirements

- level of detail varies depending on organizational needs

- **stakeholder requirements**
  - high level descriptions written from stakeholder's point of view
  - mostly expressed in natural language, informal diagrams, notation needed for problem solving (e.g., math equation)

- **system requirements**
  - detailed descriptions of system's abstract model
  - may include DFD, object class hierarchies, etc. accompanied by natural language explanations

Types of Requirements

Can we think of setting goals as good analogy for these concepts?

- **functional** (<u>what</u> system should do)
  - functions that system needs to carry out → Computer feature
  - e.g., "The system must generate a summary report on ..."
- **non-functional** (<u>how</u> system should do them)
  - quality requirements including:
    - performance
    - reliability / safety
    - usability
    - security / privacy
    - quality
    - interoperability
  - e.g., "The system must perform automated scheduled back-ups" ↳ reliability
- **environmental constraints**
  - legal issues, standards, ...
  - e.g., "The system must meet privacy regulations as stated in ..."

The example here does not clearly define 'non-functional' type of requirements. Functional & non-functional types of requirements are not so clearly explained...

Non-functional requirements

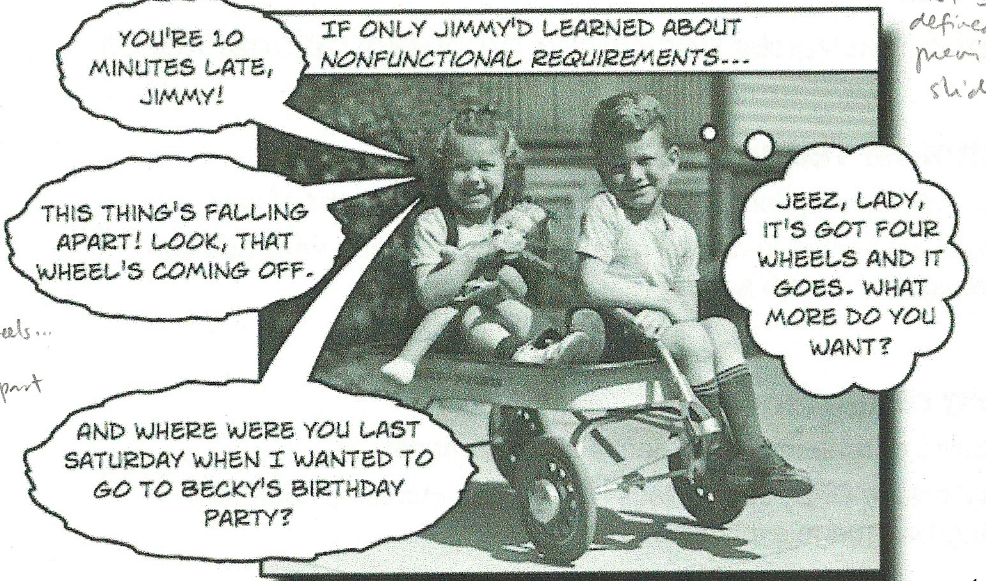
Examples of a Wagon's NFR?

acronym must be defined in previous slide...

* yes, he has met the functional requirement but missed to comply other issues of the system such as reliability of the wheels...

* is being on time part of NFR?

* what more can I get from this comics?


YOU'RE 10 MINUTES LATE, JIMMY!
IF ONLY JIMMY'D LEARNED ABOUT NONFUNCTIONAL REQUIREMENTS...
THIS THING'S FALLING APART! LOOK, THAT WHEEL'S COMING OFF.
JEEZ, LADY, IT'S GOT FOUR WHEELS AND IT GOES. WHAT MORE DO YOU WANT?
AND WHERE WERE YOU LAST SATURDAY WHEN I WANTED TO GO TO BECKY'S BIRTHDAY PARTY?

* http://www.stellman-greene.com/2009/10/03/using-nonfunctional-requirements/

This is part of Writing Requirements, eh?

Wording

- "The system must ..."
- "The system shall ..."

how to use?

- More examples:
  - "The system must be able to perform tax deduction calculations according to the CRA regulations."
  - "The system must allow the user to generate, edit, preview, print, and archive documents involving text, tables, and images."
  - "The system shall enable the user to modify object images and change object-to-class relationships easily."

Question

- "I need the car to accelerate to 80 km/hr in 10 seconds or less"

- what type of requirement is this?
  a. a functional requirement
  b. a legal constraint
  *c. a performance requirement ✓
  d. a quality requirement
  e. A and C ← this is my answer...
  what the system should do & how it should do it...

This ACRONYM should be defined on previous slide...

RE Process

- set of activities aiming to derive, validate, maintain the requirements document

- should indicate:
  - activities to be carried out
  - structure/schedule of them
  - who is responsible for each activity
  - inputs and outputs of each activity
  - tools used in each activity (if any)

RE Activities

- **requirements elicitation**
  - discovering requirements through consultation with stakeholders and research on relevant documents in the domain and market

In other words, knowing the requirements...

- **requirements analysis and negotiation**
  - analyzing elicited requirements and negotiating with stakeholders to determine which ones will be accepted (i.e., implemented)
- **requirements verification and validation**
  - checking requirements for consistency, completeness, and meeting user needs
- **requirements management**
  - supporting the changes made to existing requirements

Recall Why is Requirements Elicitation Difficult from Lec 4:

- users only think in terms of the environment they know
- requirements for new systems and new designs are always stated in current environment's terms
- how should users express what they want, instead of what they don't like?
- requirements are often unstable
- what we think we like ≠ what we actually like

Requirements Elicitation

- problems of scope
  - establishing system boundaries
- problems of understanding
  - what the customer wants
- problems volatility
  - changing requirements (within a project)

because of these problems we have systematic methods to mitigate them

Elicitation Methods

- questionnaire
- interviews
- brainstorming
- focus groups
- mockups and prototyping
- ethnographic study
- joint requirements planning
  - analog to joint application design (JAD)
- use case development
  - use case diagrams in UML

↑ another undefined acronym

Question Types

- context free questions
  - about the nature of project, environment, user profile
  - e.g., how is success measured?
  - e.g., who is the user?
- open ended questions
  - encourages full, meaningful answer
  - uses subject's own knowledge
  - gets general idea
- closed ended questions
  - have limited options as answer: Yes/No, short words
  - gets specific idea or confirmation

Interview Template

1. assess the problem

2. establish customer and user profile

3. understand user profile and usage environment

4. recap for understanding

Questions To Ask

- identifying overall purpose
  - why are we building this system?
  - what do you expect from it?
  - who are some/other users of this system?

- what is the goal of asking these questions?

⇒ To get a general overview or perspective of the system...

⇒ To establish the skeleton of the project

Questions To Ask (cont.)

- service issues
  - what kinds of services do you need from software?
  - how should these services be provided?
  - how do you currently carry out these tasks?
    (automation hints)
  - who are your customers?
  - what kinds of services do your customers need from the software?
  - how should these services be provided to your customers?
  - does the system need to operate with any other systems?
    - who else can tell me about the details of these systems?

- what is the goal of asking these questions?

⇒ To understand how system should do the requirements...

Questions To Ask (cont.)

- information and data issues
  - what kind of information or data would be useful in decision making?
  - what kind of analysis or summary reports would you like to see?
  - who should have (read) access to this information?

- what is the goal of asking these questions?

⇒ To ensure privacy and security on the side of the user...

⇒ To satisfy the stakeholders

Questions To Ask (cont.)

- performance issues
  - what specific tasks must the system perform?
    - identify **throughput** and **response time** for each task
  - are any of these tasks more crucial than others?

- what is the goal of asking these questions?

⇒ To ensure reliability of the system & ~~all~~ specific requirements are met...

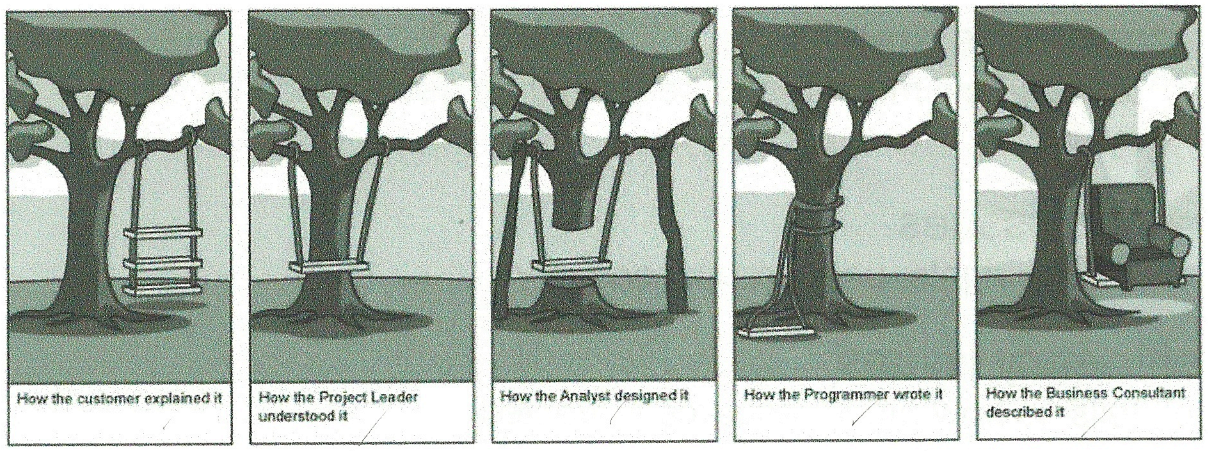
Questions To Ask (cont.)

- economy issues time, money, staff
  - tells us how much resource should be devoted to the system
- control issues
  - access rights, user groups, security
- efficiency issues
  - tells us how much resource is wasted
- usability issues
  - types of people using the system
  - examples of what they (didn't) like from past experience

How to document requirements?

- bullet points of text?
- diagrams?
- other methods?
- which is better when?

Importance of Documentation


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it

bad communication

fancy design

bad management

how to sell it

partial image taken from: http://www.sa-depot.com/?p=203

what does this mean?

UNIFIED MODELING LANGUAGE

UML Structural Modeling

- diagrams that capture static system behaviour
  - relationships among objects (e.g., classes, use cases, states, components)
- diagrams involved:
  - class diagram
  - object diagram
  - deployment diagram
  - package diagram
  - component diagram
- objects in these diagrams represent the elements of the system and mechanism to assemble them

UML Behavioural Modeling

- diagrams that capture dynamic behaviour - when system is running
  - stream of input events
  - system reactions
- diagrams involved:
  - use case diagram
  - sequence diagram
  - collaboration diagram
  - state chart diagram
  - activity diagram

Use Cases

- considered as part of high level requirement analysis activity
- used to gather requirements
  - focus on system functions
  - identify actors? identify factors
  - capture internal and external influences
  - show interaction between actors and system
- used to provide an outside view of system
- use cases do not:
  - specify UI design
  - specify non-functional requirements
  - specify implementation detail

Use Case Diagrams

- define a set of use cases, actors, and their relationships
    - **use case** = collection of possible scenarios between the system and actors showing how the primary actor's goal might be delivered or might fail
    - **actor** = internal or external controller
        - human user
        - internal application
        - external application
    - **primary actor**: use system to achieve a goal
    - **secondary actor**: actors that system needs assistance from to achieve the primary actor's goal

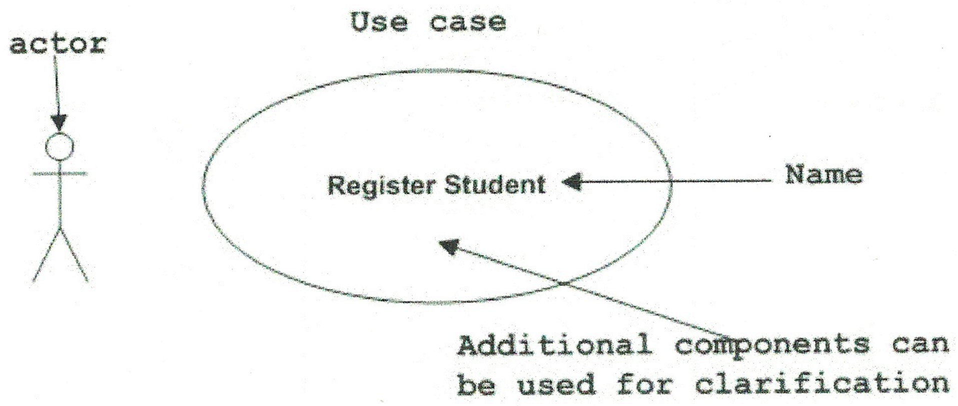
Developing Use Cases

- choose system boundary
- identify primary actors of the system
- list each actor's goals in using the system
- structure a use case around each primary actor's goal
    - e.g.: pay bill, register for course, place order, track progress, monitor productivity
- scenario shows sequence of steps to achieve goal
    - each step may be a subgoal which represents another use case or an autonomous action

Tips

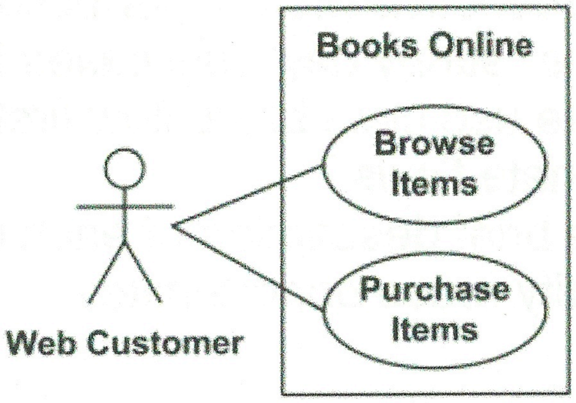
- be productive without perfection
- define "sunny day" use cases first
- create use case basic flow first, then alternate flows
- have brief description of each use case
- identify reuse opportunities

Notation



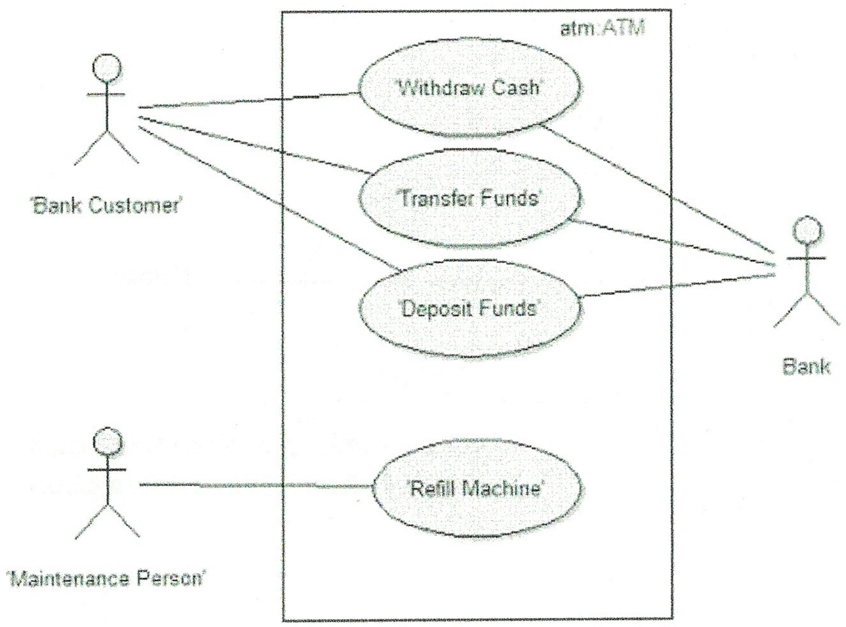
Example: Ordering Books Online

- simple example:



- presumably, actor also has to register/sign in, specify shipping options

Example: ATM transactions



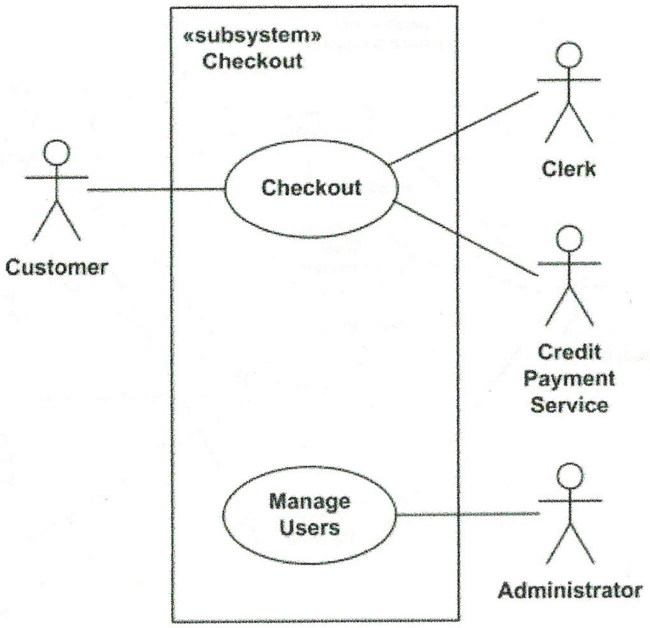
Use Case Relationships

- **generalization** - abstract, non-detailed use case can generalize more specific use cases
  - e.g.:Verify Identity generalizes Check Fingerprint and Check RFID tag
- **inclusion** - factor out common functionality as separate use case to be included by others
  - factored out use case is typically not stand alone
- **extension** - plug in additional functionality to a base use case (like inheritance)
  - can specify conditions for extended use case

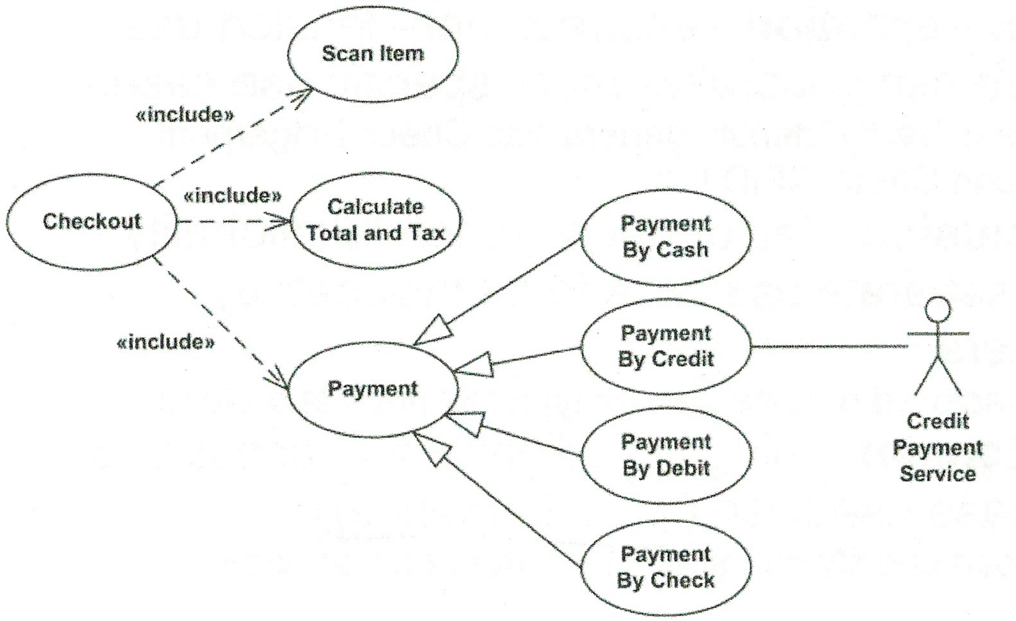
I am confused with these 2 relationships

Example:
Point of Sale Terminal Checkout

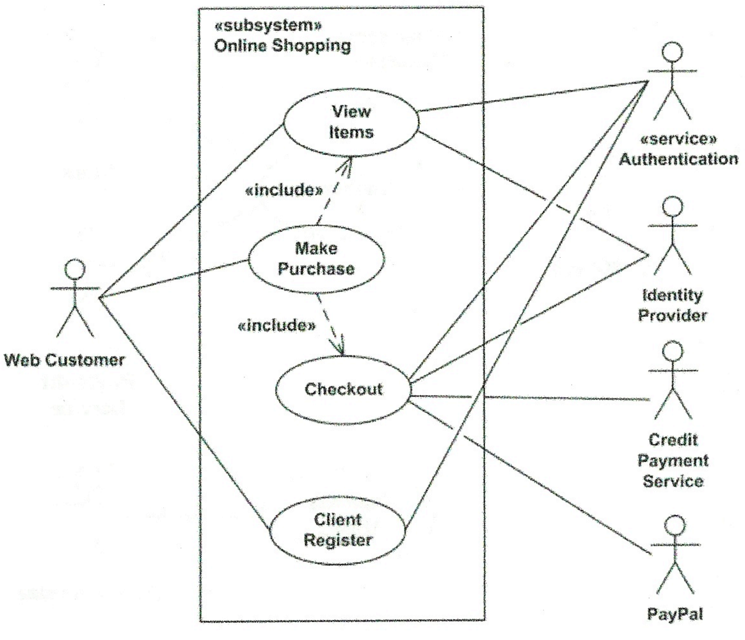
e.g.,
Superstore
self-checkout



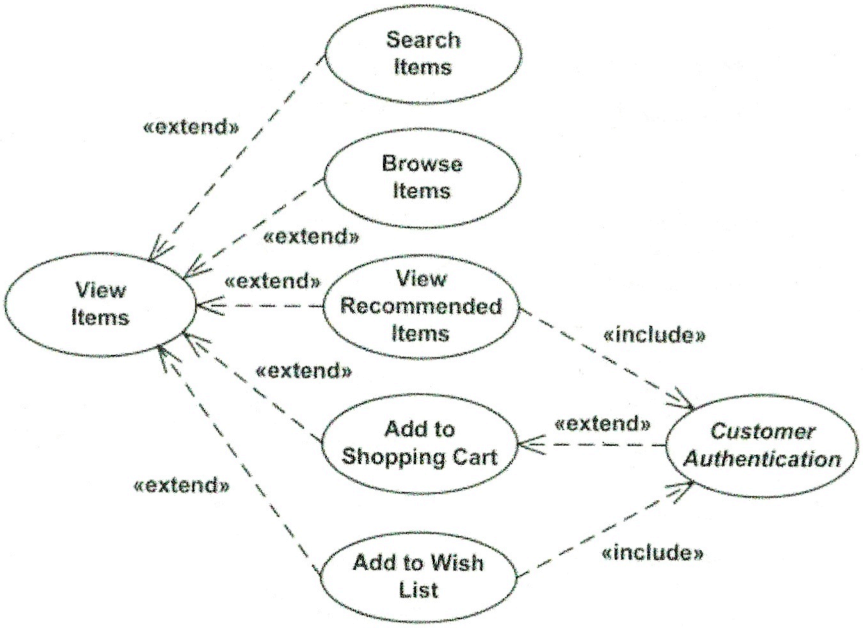
Checkout in detail



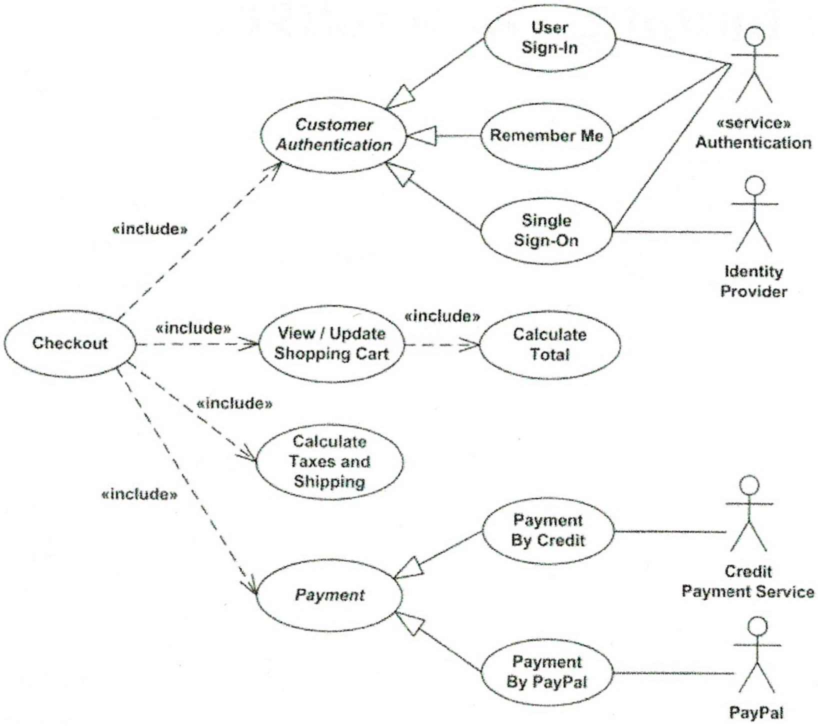
Example:
Online Shopping



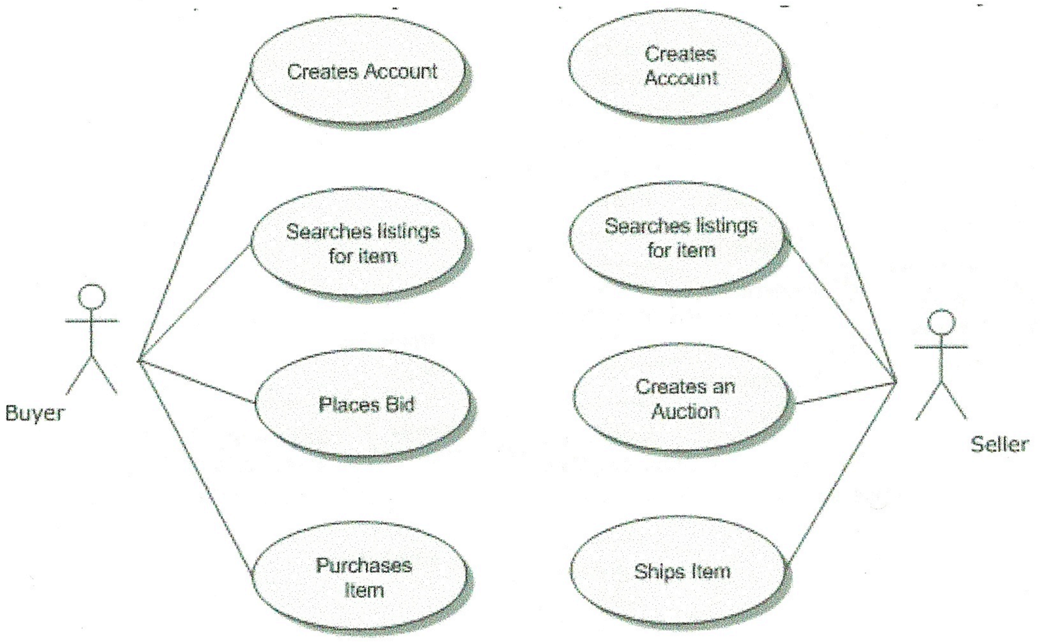
View items in detail



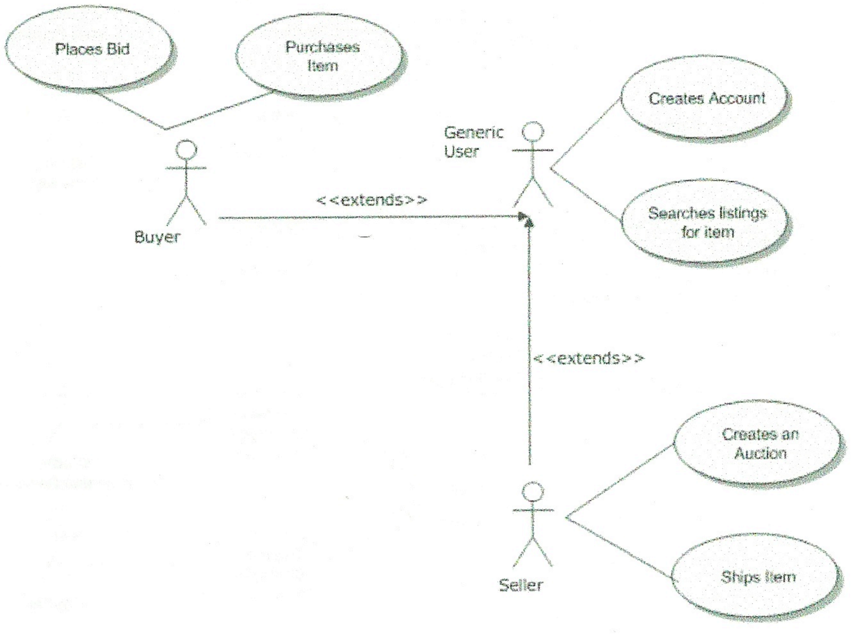
Checkout in detail



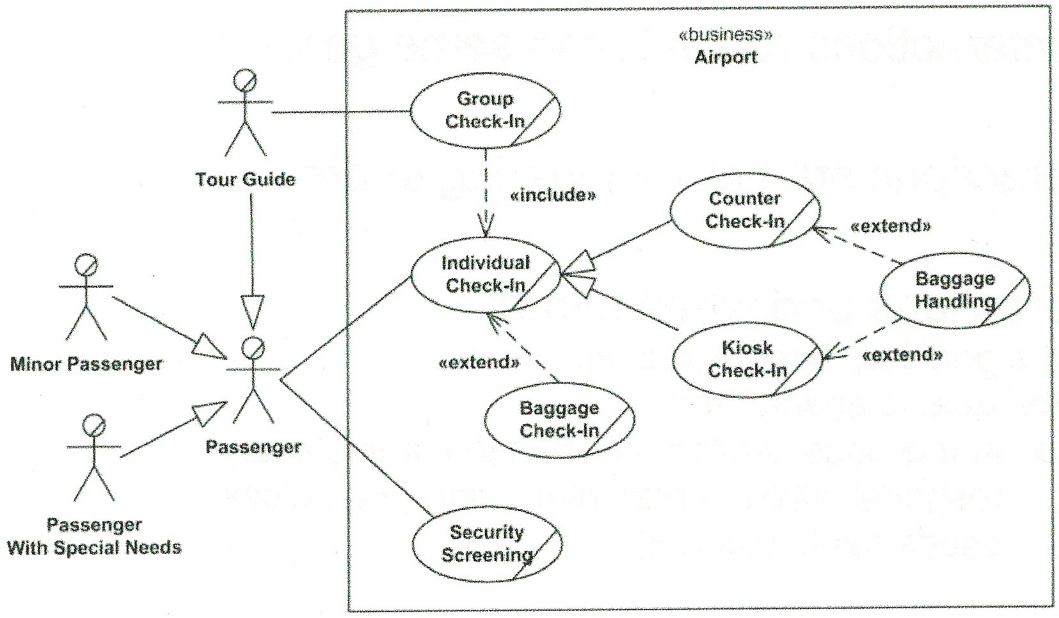
Example: Ebay bidding



Example: Ebay bidding (with reuse)



Example: Airport check-in



How to ...

- order pizza online from Domino's?
- upload a picture to your Facebook account?
- compare two products on Bestbuy's?

Scope of Use Cases

- all interactions relate to the same goal
- interactions start at a triggering event
- interactions end when either:
    - the goal has been achieved
    - the goal is abandoned
        - in this case, system still has to complete its responsibilities in that interaction (e.g., close user's bank account)

Applicability

- the system being modeled can be:
    - a computer system - shows how humans and computer system interact
    - a component within a system - shows behaviour of a single component and how it operates with other components or external actors
    - a system of systems - show complex interaction among these systems and the outside world
    - an organization - shows either how the organization interacts with outside world or shows interaction among organization's internal processes

Documenting Requirements

- ranking of importance
  - essential, conditional, optional
- verifiability
  - can users read and attest to the requirement?
  - can each requirement be tested?
- modifiability
  - state possible changes and probability
- traceability
  - using a matrix to relate associate business rules and test cases

Requirements Traceability

- method for linking requirements to sources and business operation rules
- a traceability matrix consists of:
  - associations between
    - business rules
    - use cases
    - requirement items
  - test data for each item
- result: matrix shows specific requirement deliverables

Example

Req #	Name	Business Rule	Project Task	Test Case	Verification
1	Calculate Interest	CGA 001	5.1.1, 6.2.1	TS 001, TS 025	Yes / No
2	...	...	...	...	...

- Req #: Requirement number in document
- Name: Requirement item name/description
- Business rule: Associated business rule
- Project task: Associated project task number
- Test case: Test cases prepared for testing requirement
- Verification: Record of completion in signoff process

Requirements Analysis and Negotiation

- analysis:
  - determine cost of requirements
  - identify dependencies
  - finalize scope and limitations
- negotiation:
  - resolve conflicts between requirements
  - discuss issues with stakeholders
  - priority and acceptance creation
- importance:
  - avoiding scope/cost/schedule creep
  - avoiding conflicts
  - avoiding ambiguous requirements
  - understand customer needs

Requirements V & V

- verification (building the thing right):
  - inspections
    - do we all interpret requirement the same way?
    - are the requirements complete?
  - metrics
    - % of requirements inspected
    - # requirements reviewers interpreted the same
      total # requirements reviewed
  - configuration management
    - central repository
    - strict version control
- validation (building the right thing):
  - best to involve stakeholders (especially end-users)

Requirements Management

- requirements identification

- a change management process
  - documenting critical incidents

- traceability policy

- CASE tool support:
  - spreadsheets
  - simple DB systems

Exercise

- scenario:
    - "Write me an app that determines the most popular professor in the university."
- who are the stakeholders?
- what will you ask?
- how will you obtain test cases?