Requirements Engineering
(II)

1

RE Tasks

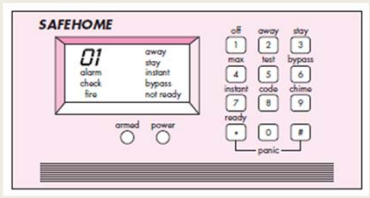
- Inception
- Elicitation
- Elaboration
- Negotiation
- Specification
- Validation
- Requirements management

Eliciting Requirements

- Meetings are conducted and attended by both software engineers and customers
- Rules for preparation and participation are established
- An agenda is suggested
- A "facilitator" (can be a customer, a developer, or an outsider) controls the meeting
- A "definition mechanism" (can be work sheets, flip charts, or wall stickers or an electronic bulletin board, chat room or virtual forum) is used
- The goal is
  - to identify the problem
  - propose elements of the solution
  - negotiate different approaches, and
  - specify a preliminary set of solution requirements

3

Example: SafeHome

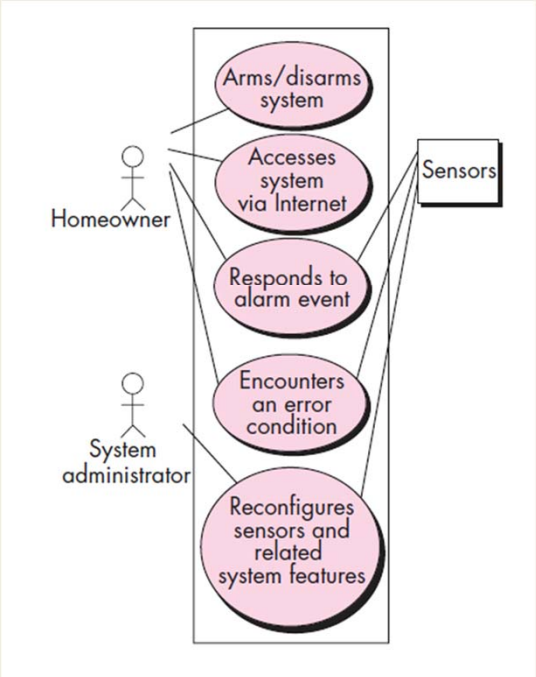


Example use case

- **Use case:** *InitiateMonitoring*
- **Primary actor:** Homeowner.
- **Goal in context:** To set the system to monitor sensors when the homeowner leaves the house or remains inside.
- **Preconditions:** System has been programmed for a password and to recognize various sensors.
- **Trigger:** The homeowner decides to “set” the system, i.e., to turn on the alarm functions.
- **Scenario:**
  - 1. Homeowner: observes control panel
  - 2. Homeowner: enters password
  - 3. Homeowner: selects “stay” or “away”
  - 4. Homeowner: observes read alarm light to indicate that *SafeHome* has been armed

4

Use-Case Diagram



5

UML use case diagram for *SafeHome* home security function

Elaboration: Building the model

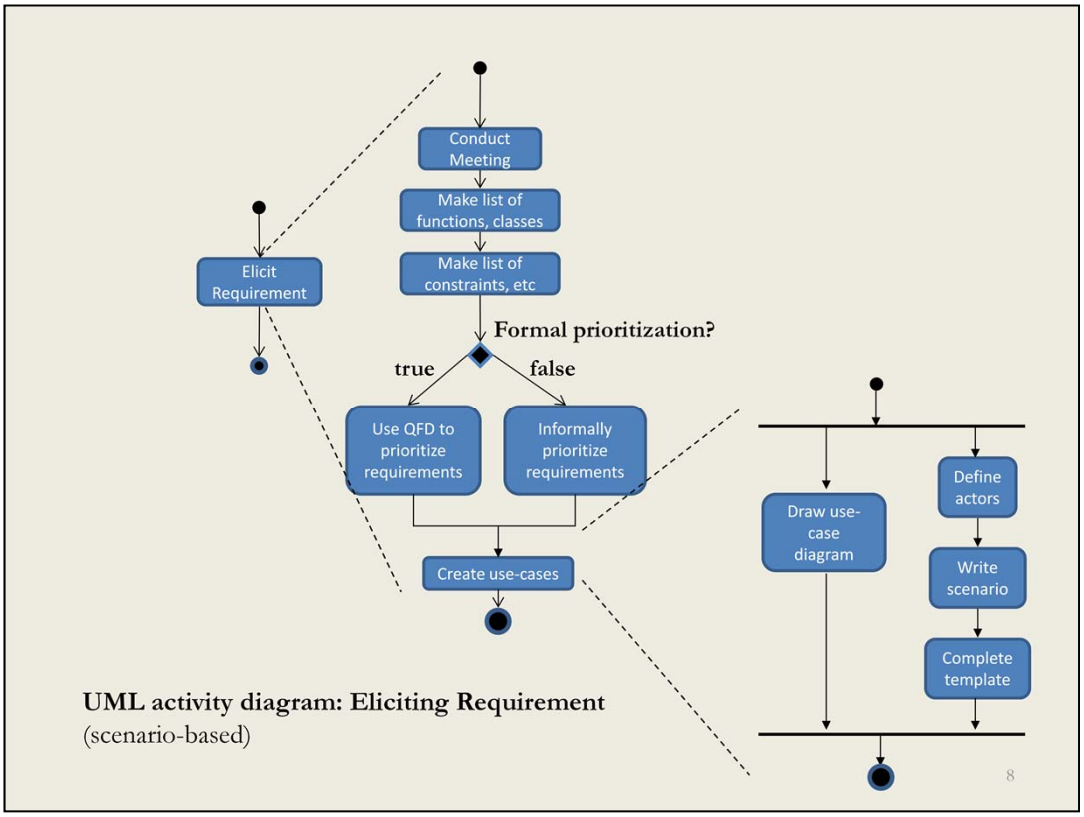
- The analysis/requirement model
  - To provide a description of the required informational, functional, and behavioral domains for a computer-based system.
  - The model changes dynamically as you learn more about the system to be built
  - The analysis model is a snapshot of requirements at any given time. It is expected to change.
- The most common elements to be developed
  - Scenario-based elements
  - Class-based elements
  - Behavioral elements

6

RE model versus process model

Scenario-based elements

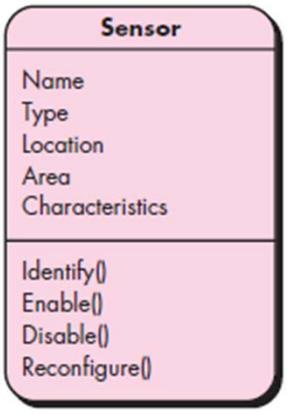
- The system is described from the user's point of view using a scenario-based approach
  - (E.g., basic use cases and their corresponding use-case diagrams evolve into more elaborate template-based use cases.
- Represented by activity diagram
- Serve as input for the creation of other modeling elements.



UML activity diagram: Eliciting Requirement
(scenario-based)

Class-based elements

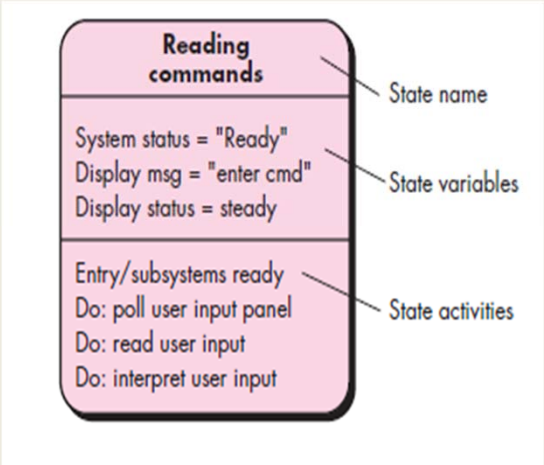
- Each usage scenario implies a set of objects that are manipulated as an actor interacts with the system.
- These objects are categorized into classes
- Represented by class diagram



9

Behavioral-based elements

- The behavior of a computer-based system can have a profound effect on the design that is chosen and the implementation approach that is applied.
- The *state diagram* is one method for representing the behavior of a system



10

A *state* is any externally observable mode of behavior. In addition, the state diagram indicates

actions (e.g., process activation) taken as a consequence of a particular event.

Negotiation

- Goal
  - To develop a project plan that meets stakeholder needs while at the same time reflecting the real-world constraints (e.g., time, people, budget) that have been placed on the software team.
- The best negotiations strive for a “win-win” result
- A set of negotiation activities
  - Identify the key stakeholders
    - These are the people who will be involved in the negotiation
  - Determine each of the stakeholders “win conditions”
    - Win conditions are not always obvious
  - Negotiate
    - Work toward a set of requirements that lead to “win-win”

11

In an ideal case, the inception, elicitation, and elaboration tasks determine customer requirements in sufficient detail to proceed to subsequent software engineering activities. Unfortunately, this rarely happens.

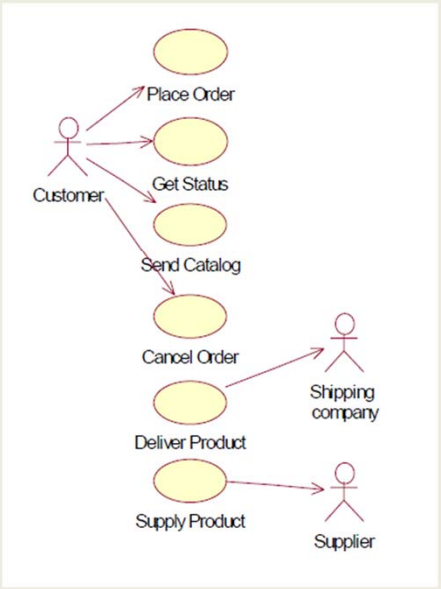
Validating requirements

- As each element of the requirements model is created, it is examined for inconsistency, omissions, and ambiguity.
- Guiding questions
  - Is each requirement consistent with the overall objective for the system/product?
  - Have all requirements been specified at the proper level of abstraction?
  - Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
  - Is each requirement bounded and unambiguous?
  - Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
  - Do any requirements conflict with other requirements?
  - …..

12

Use Case Diagram

- Created to visualize the interaction of your system with the outside world.



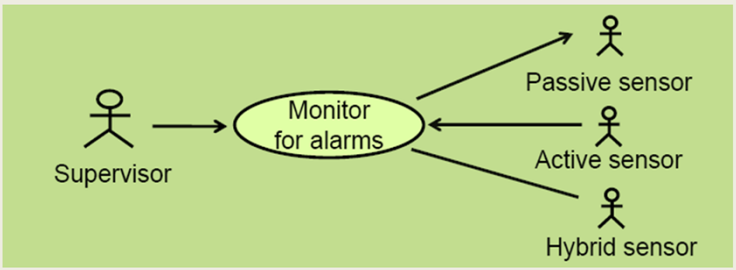
13

Use Case Diagram (Cont.)

- Actor
  - a role that a human, hardware device, or another system can play in relation to the system.
  - Brief name
- Use Case
  - Use the active tense; begin with a verb
  - Indicate the value or goal of the actor

Use Case Diagram (Cont.)

- Relationship (between actors and use cases)
  - A line is used to represent a communicates-association.
  - An arrowhead indicates who initiates each interaction.
  - No arrowhead indicates either end can initiate each interaction.



15

Practice

- Youtube
  - four use cases
    - Watch cat video (e.g., the *keyboard cat*)
    - Upload cat video
    - Search for cat video
    - Share cat video with friends
- Facebook
- Find the actors and use cases for these two websites

16

Checkpoints for Actors

- Have you found all the actors?
  - Have you accounted for and modeled all roles in the system's environment?
- Is each actor involved with at least one use case?
  - Can you name at least two people who would be able to perform as a particular actor?
- Do any actors play similar roles in relation to the system?
  - If so, merge them into a single actor.

Identify Use Cases

- What are the goals of each actor?
  - Why does the actor want to use the system?
  - Will the actor create, store, change, remove, read data in the system? If so, why?
  - Will the actor need to inform the system about external events or changes?
  - Will the actor need to be informed about certain occurrences in the system?
- Does the system supply the business with all of the correct behaviors?

18

Checkpoints for Use Cases

- The use case model presents the behavior of the system; it is easy to understand what the system does by reviewing the model.
- All use cases have been identified; the use cases collectively account for all required behavior.
- All features map to at least one use case.
- The use case model contains no superfluous behavior; all use cases can be justified by tracing them back to a functional requirement.
- All *valueless* CRUD use cases have been removed.
  - Create, Retrieve, Update, Delete

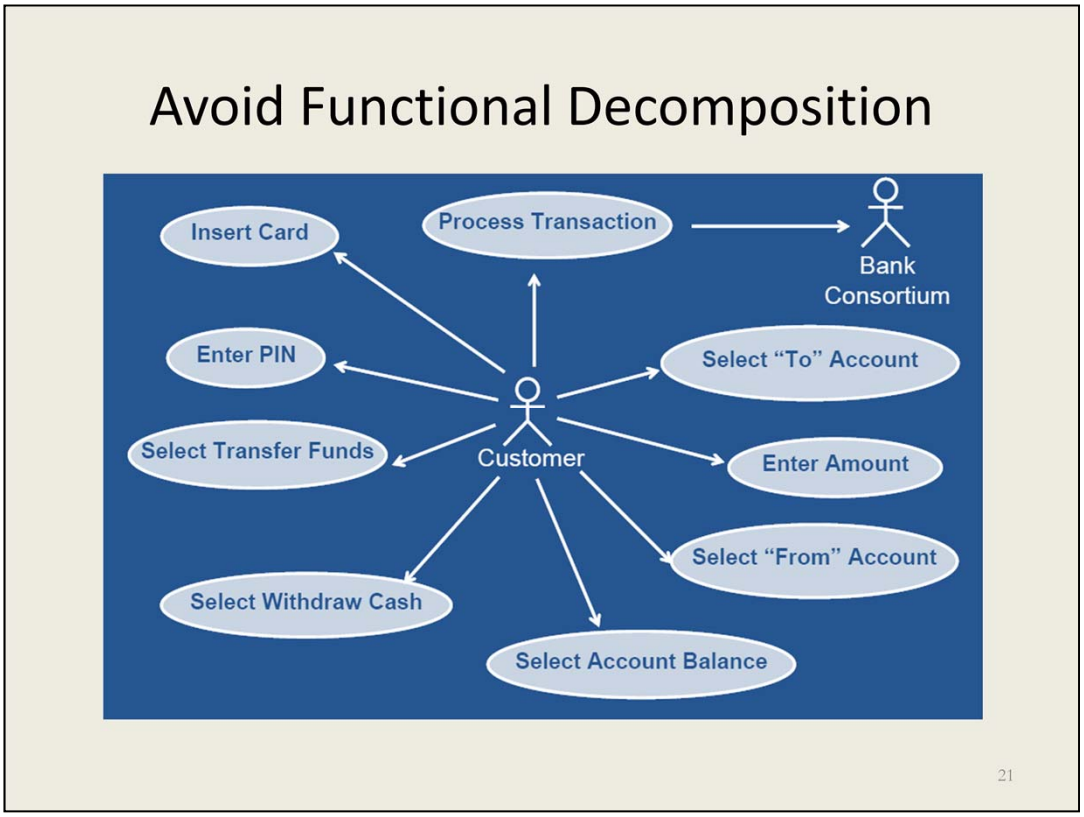
19

Remove CRUD use cases if they are data- management use cases that do not provide results that are of value to actors.

Practice - Youtube

- Actor
  - User
    - Unregistered user/Registered user
  - Advertiser
  - Content provider
  - Embedder
  - Administrator
  - Etc…
- Use cases
  - Register, Log In, Watch Video, Upload Video, Delete Video, Like, Share, Comment, Reply, Flag, Subscribe, etc…
  - Provide Commercials, etc…
  - Upload Video, Set Price, etc…
  - Embed Video, etc…
  - Block User, Delete Comment, etc…

20



It is not uncommon that the use-case model degenerates into a functional decomposition of the system. To avoid this, watch for the following symptoms:

"Small" use cases, meaning that the description of the flow of events is only one or a few sentences.

"Many" use cases, meaning that the number of use cases is some multiple of a hundred, rather than a multiple of ten.

Use-case names that are constructions like "do this operation on this particular data" or "do this function with this particular data". For example, "Enter Personal Identification Number in an ATM machine" should not be modeled as a separate use case for the ATM machine, since no one would use the system to do just this. A use case is a complete flow of events that results in something of value to an actor.

To avoid functional decomposition, you should make sure that the use-case model helps answer questions like:

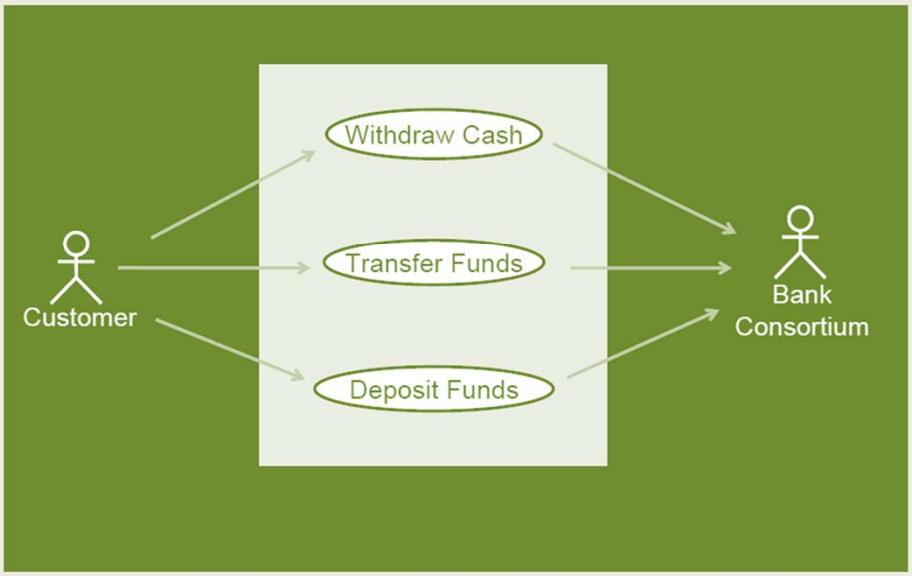
What is the context of the system?

Why is the system built?

What does the user want to achieve when using the system?

What value does the system add to the users?

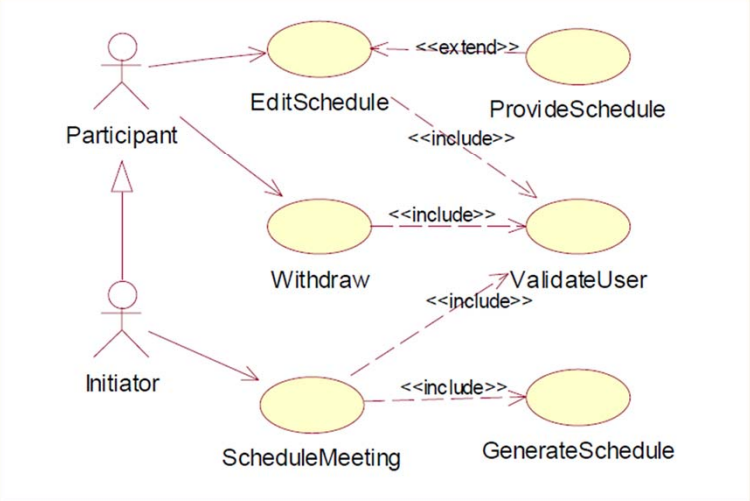
Avoid Functional Decomposition



22

Relationship Between Actors

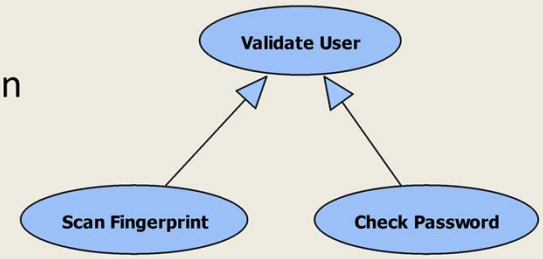
- Inheritance (Specialization / Generalization)



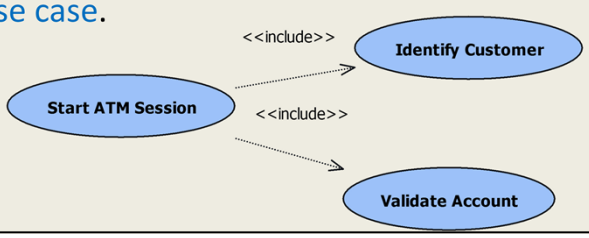
23

Relationships between Use Cases

- Generalization



- Include
  - A relationship from a base use case to an inclusion use case, specifying how the behavior defined for the inclusion use case can be inserted into the behavior defined for the base use case.



24

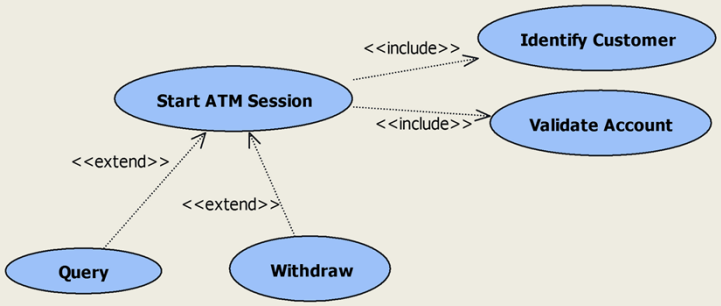
child use case inherits the behavior and meaning of the **parent use case**;

The **child** may add to or override the behavior and meaning of the **parent** use case;

The **child** may be substituted any place the **parent** appears.

Relationships Between UCs (Cont.)

- Extend
  - A relationship from an extension use case to a base use case, specifying how the behavior defined for the extension use case can be inserted into the behavior defined for the base use case.
  - Extension points have to be defined in base use case.



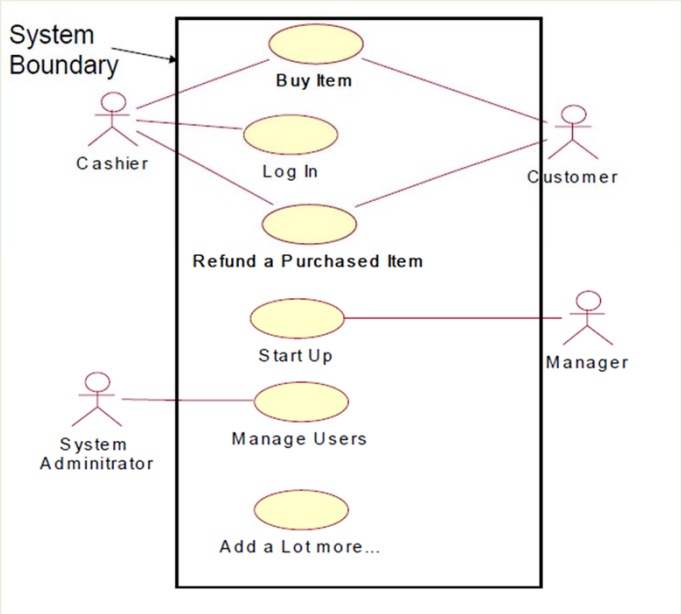
25

Extension points: transaction possible, receipt details

Include and Extend

- When to use Includes?
  - You have a piece of behavior that is similar across many use cases.
  - Break this out as a separate use case and let the others “include” it.
- When to use Extends?
  - A use case is similar to another one but does a little bit more.
  - Put the normal behavior in one use case and the exceptional behavior somewhere else.

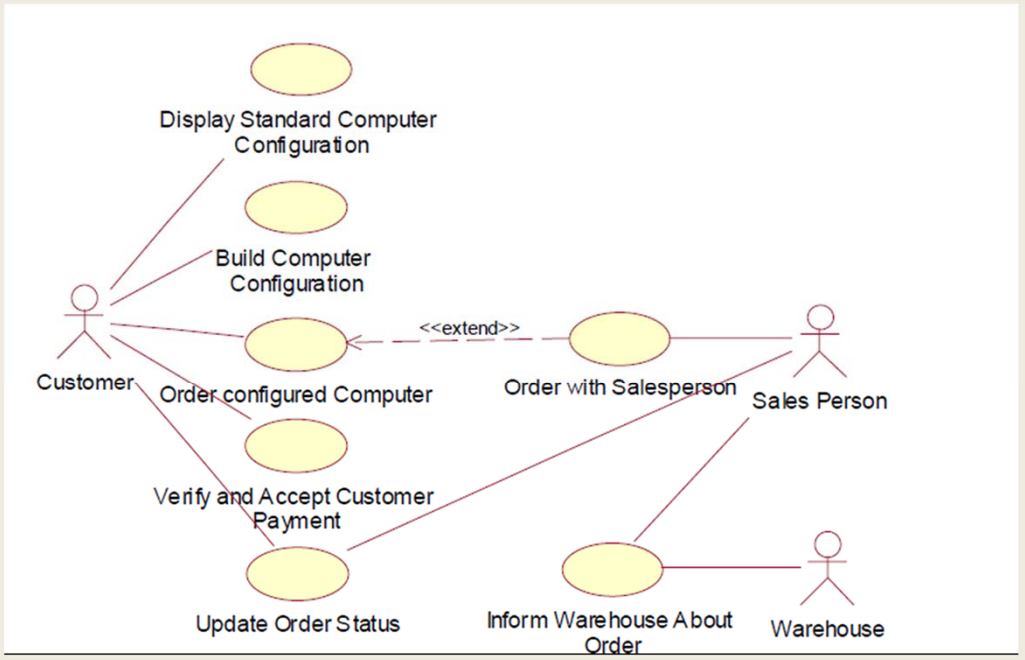
System Boundary



27

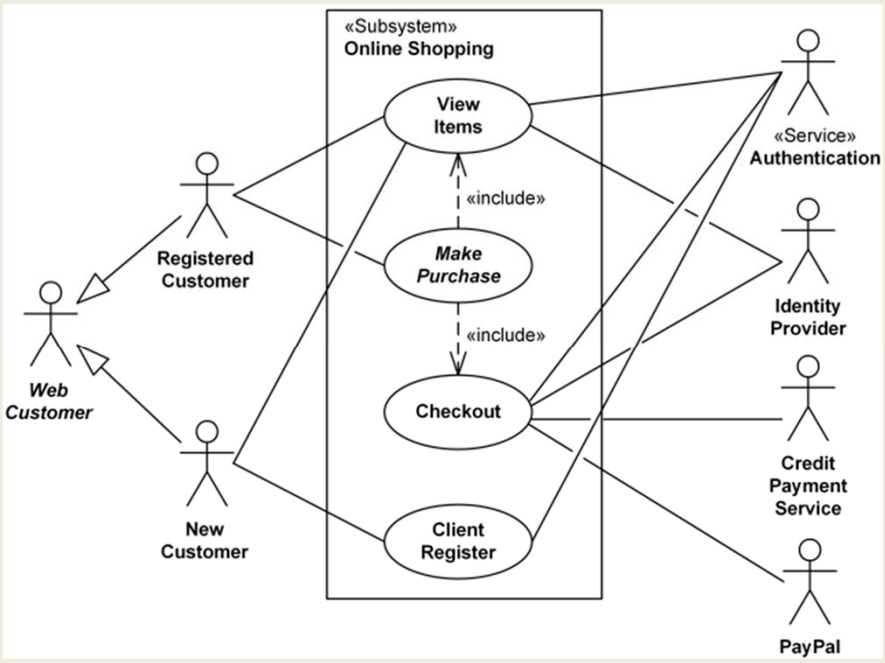
Exercise

- A computer manufacturer provides opportunities for online purchasing through Internet.
- Customers can visit their website and select a computer to buy.
- Computers are defined in three categories: server, PC, laptop.
- Customers can choose default configurations, or customize their own by selecting configurable parts (memories, hard drives, etc.) from a list. For each configuration, the system calculates the price.
- To submit an order, the customer has to provide shipping and billing information, credit card or checks are both accepted.
- Once the order has been submitted, the system sends a confirmation email to the customer with details.
- Customer can check the status of an order before it is delivered.
- The sales team manages the orders that have been submitted. The process includes the following steps: verify the billing information, send the configuration to the warehouse, print receipt, and request shipment.

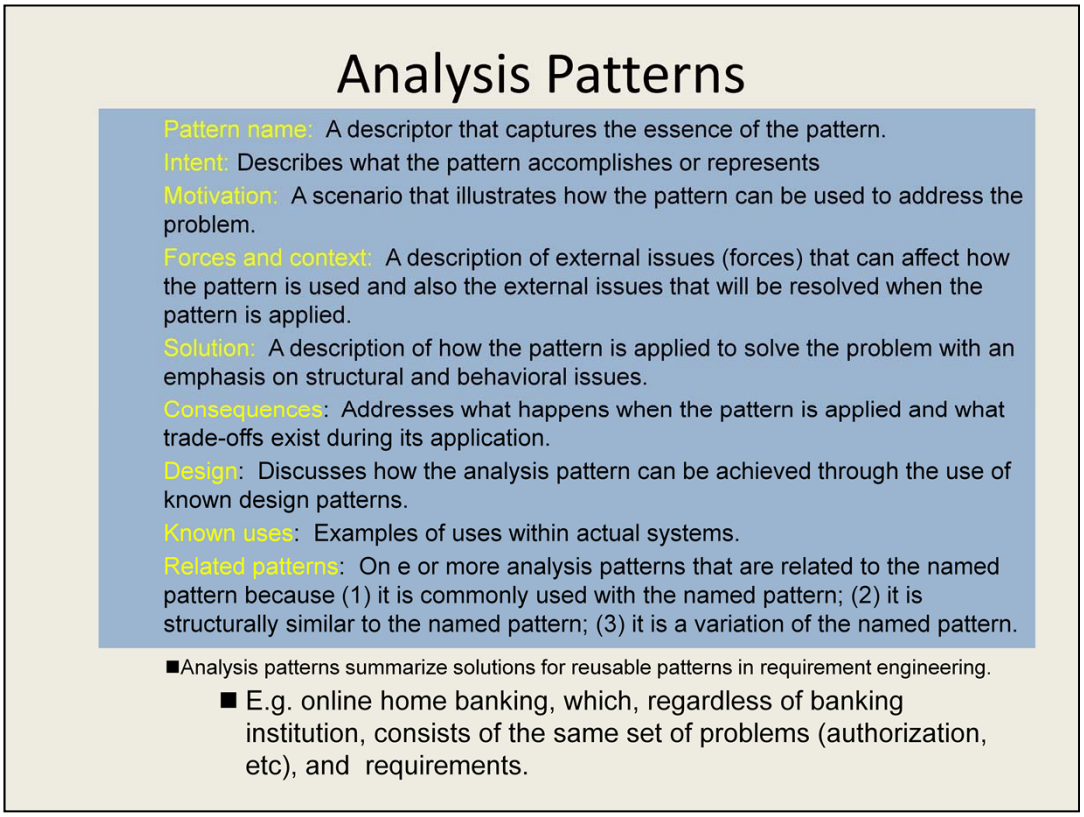


Bad use case diagram: most use cases are wrong in one way or another.

29


«Subsystem»
Online Shopping
View Items
«include»
Make Purchase
«include»
Checkout
Client Register
Registered Customer
Web Customer
New Customer
«Service»
Authentication
Identity Provider
Credit Payment Service
PayPal

30



certain problems reoccur across all projects within a specific application domain.18 These *analysis patterns* [Fow97] suggest solutions

(e.g., a class, a function, a behavior) within the application domain that can be reused when modeling many applications.

Analysis patterns are integrated into the analysis model by reference to the pattern name. They are also stored in a repository so that requirements engineers can use search facilities to find and apply them.

Useful Resource

- UML
  - Use case diagrams
    http://msdn.microsoft.com/en-us/library/vstudio/dd409427.aspx
  - Notations
    http://www.tutorialspoint.com/uml/uml_basic_notations.htm
- Use case development
  - Use case identification
    http://www.upedu.org/process/gdlines/md_ucmod.htm
  - CRUD
    http://businessanalystlearnings.com/ba-techniques/2013/3/13/techniques-for-identifying-use-cases

Summary

- RE Tasks
  - Elaboration: requirement modeling
  - Negotiation
  - Specification
  - Validation
  - Management
- Use case diagrams (UML)
  - Actors and use cases
    - Identification (checkpoints)
    - Relationships
  - Examples