Laboratorio di Tecnologie dell'Informazione

Ing. Marco Bertini
bertini@dsi.unifi.it
http://www.dsi.unifi.it/~bertini/

Software engineering techniques

"A fool with a tool is still a fool."
- Grady Booch

Use case

- A use case in software engineering and systems engineering is a description of a system's behavior as it responds to a request that originates from outside of that system.
- In other words, a use case describes "who" can do "what" with the system in question.
- Use cases describe the system from the user's point of view.
- Each use case focuses on describing how to achieve a **goal** or task.

Use case - cont.

- Each use case should convey a primary scenario, or typical course of events, also called "basic flow", "normal flow," "happy flow" and "main path". The main basic course of events is often conveyed as a set of usually numbered steps.
- Alternate paths can be written, e.g. next to the steps of the main path.

Use case: example

Main path

1. The system prompts the user to log on,

2. The user enters his name and password

3. The system verifies the logon information

4. The system logs user on to system

Alternate path

2.1 The user swipes an RFID card on a reader

Use case - cont.

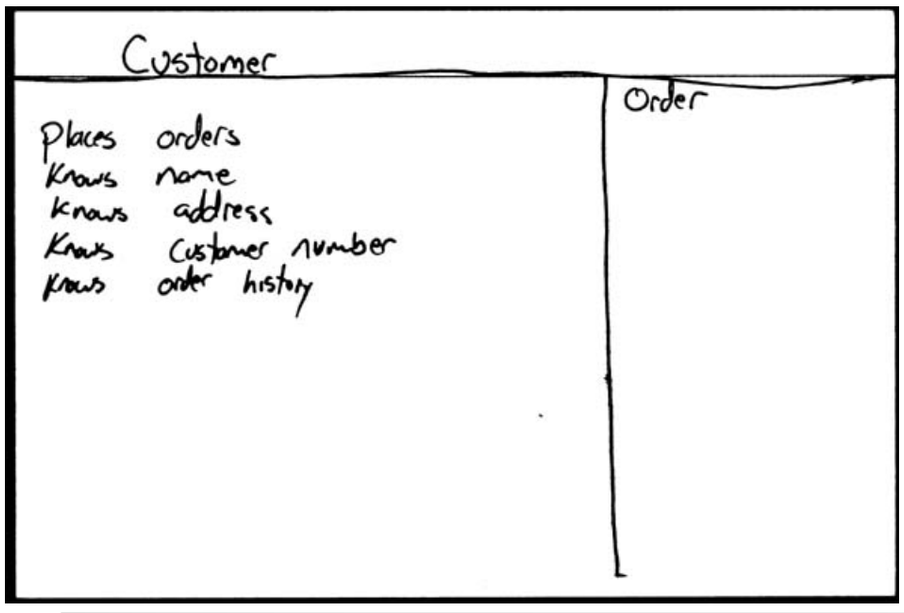
- Reread the use case, check it focuses on getting the task done.
- Pay attention to the nouns in the use case: they are candidates to identify the classes needed to model the system, and tell what to focus on
- Look at the verbs: they are candidates to identify the methods of the classes

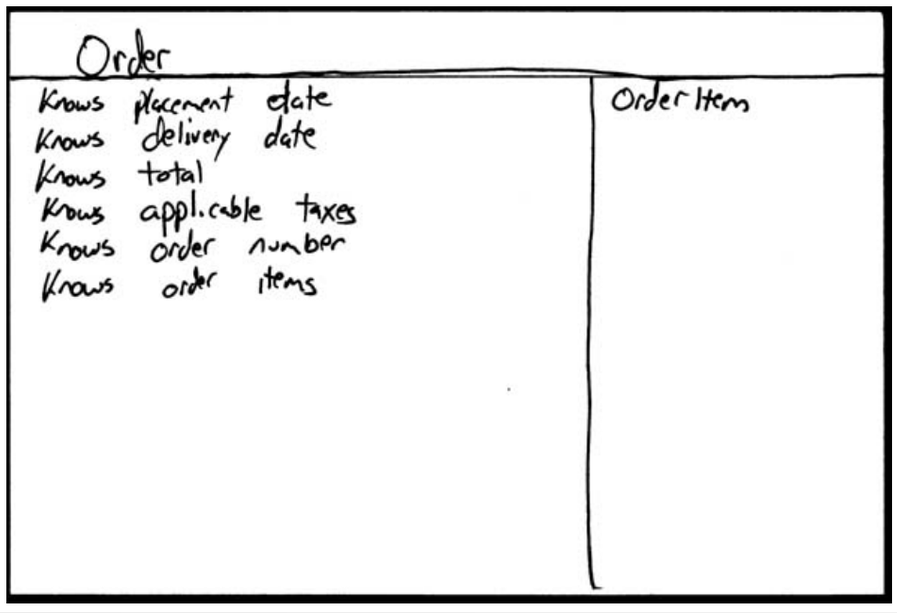
CRC cards

- CRC cards (Class, Responsibility, Collaborator) are a technique for discovering object classes, members and relationships in an object-oriented program.
- A **class** represents a collection of similar objects, a **responsibility** is something that a class knows or does, and a **collaborator** is another class that a class interacts with to fulfill its responsibilities.

CRC cards

Class Name	
Responsibilities	Collaborators





CRC cards - cont.

- To create CRC classes iteratively perform the following steps:

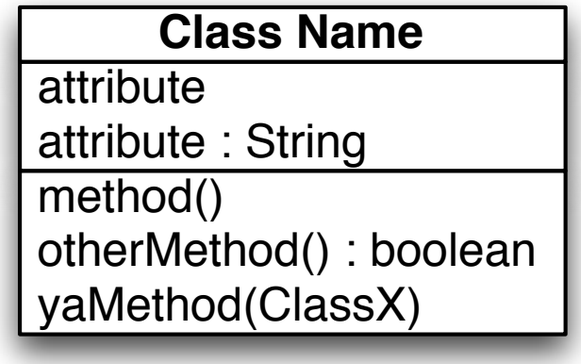
1. Find classes: look for the three-to-five main classes
2. Find responsibilities: ask yourself what a class does as well as what information you wish to maintain about it.
3. Define collaborators: a class often does not have sufficient information to fulfill its responsibilities. Therefore, it must collaborate (work) with other classes to get the job done: requesting info or to perform a task
4. Move the cards around: it's a method to understand the system: classes that collaborate should stay next each other

UML Class diagram

- A UML class diagram describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.
- They are being used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code.

UML Class diagram - cont.

- In the class diagram these classes are represented with boxes which contain three parts:
- The upper part holds the name of the class
- The middle part contains the attributes of the class (and their type)
- The bottom part gives the methods or operations the class can take or undertake



UML Class diagram - cont.

- In the conceptual design of a system a number of classes are identified and grouped together in a class diagram, which helps to determine the statical relations between those objects. With detailed modeling the classes of the conceptual design are often split in a number of subclasses.
- There can be several different types of relations among the classes, drawn as lines and arrows

UML Class diagram: example

- Several tools allow to generate code from UML class diagrams, or reverse engineer code to UML class diagrams