Université de Paris

Génie Logiciel
UML to model requirements – part 2

Sylvain Lobry

21/10/2022

UML to model requirements

Types of diagrams

- UML defines 13 diagrams in 3 categories which can define a system according to different points of view
- Structure diagrams
    - Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram and Deployment Diagram
- Behavior diagrams
    - Use Case Diagram, Activity Diagram and State Machine Diagram
- Interaction diagrams
    - Sequence Diagram, Communication Diagram, Timing Diagram and Interaction Overview Diagram

UML to model requirements

Types of diagrams

- UML defines 13 diagrams in 3 categories which can define a system according to different points of view
- Structure diagrams
  - Class Diagram, Object Diagram, Component Diagram, Composite Structure Diagram, Package Diagram and Deployment Diagram
- Behavior diagrams
  - Use Case Diagram, Activity Diagram and State Machine Diagram
- Interaction diagrams
  - Sequence Diagram, Communication Diagram, Timing Diagram and Interaction Overview Diagram

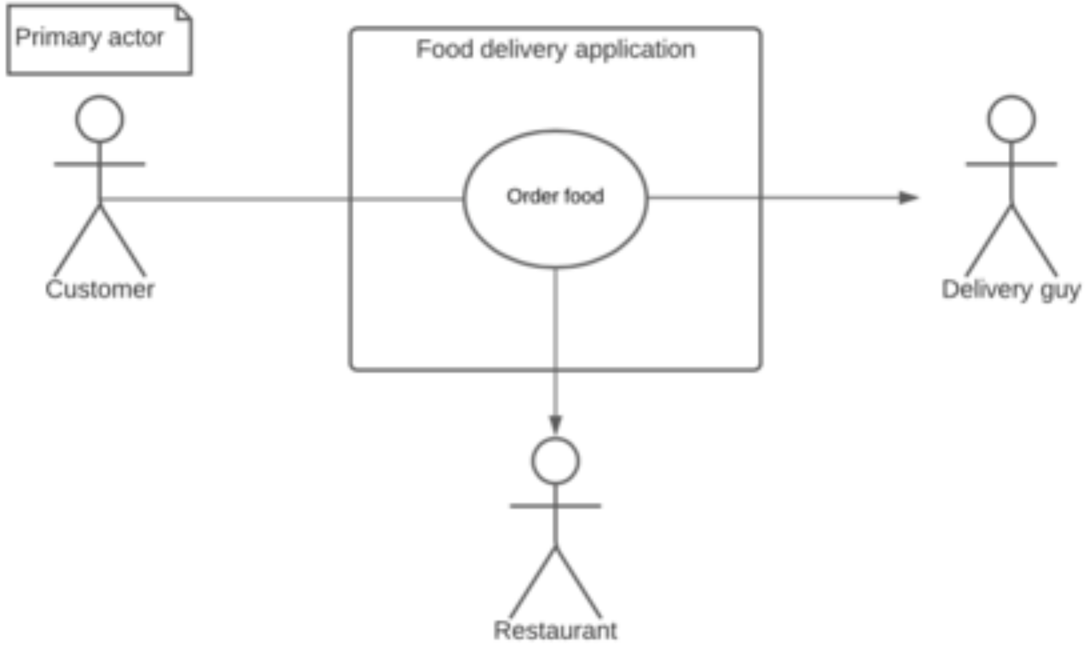
Use case diagrams

Use case

- Definition: A *use case* is the specification of a set of actions performed by a system, which **yields an observable result** that is, typically, of value **for one or more actors or other stakeholders of the system**. (OMG UML Superstructure 2.0)

Use case diagrams

A simple use case diagram

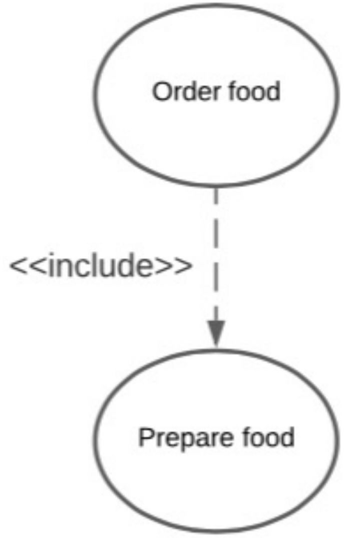


Use case diagrams

Relations between use cases - Inclusion

- So far, we have seen actors, use cases and relations between them.
- It is also possible to have relations between use cases

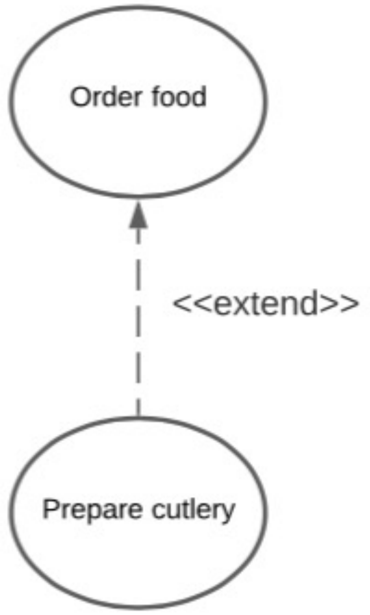
- Inclusion relation
- represented by a dashed arrow with the specialization <<include>>
- Can describe a sub-functionality
- Or can be used to share functionalities
- Must **always** be ran
- Does not directly answer an objective from primary actor



Use case diagrams

Relations between use cases - Extension

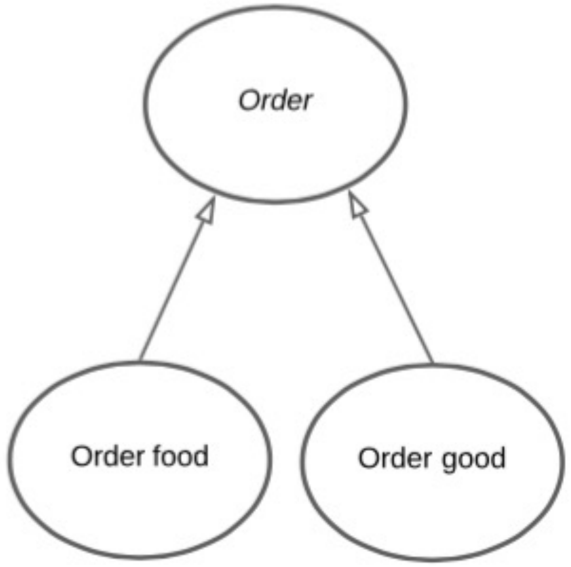
- Extension: similar to inclusion, but **optional**
- Application of an extension is decided during the scenario.
- Represented by a dashed arrow with the specialization <<extend>>



UML to model requirements

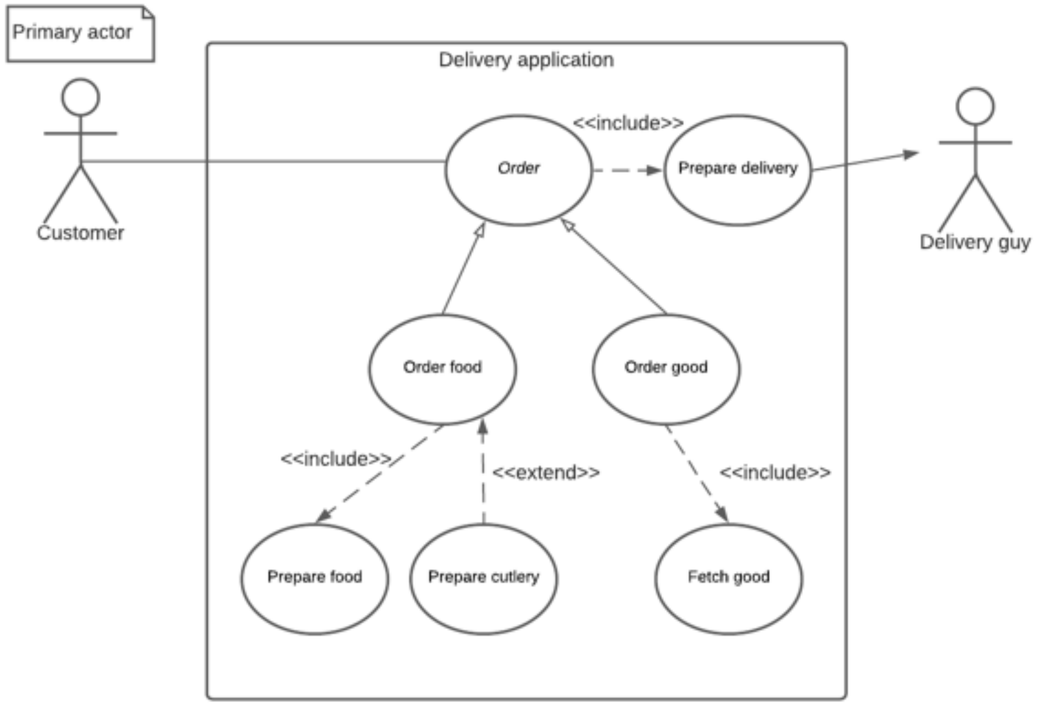
Relations between use cases

- Specialization: as for classes.
- Gives a sub use case
- Allows to inherit the behavior, the associations to actors and use cases
- The use case it generalizes from is often abstract. In which case, the name is in *italic*.
- Representation: white arrow.



Use case diagrams

A less simple use case diagram



UML to model requirements

How to represent a use case?

- Use case diagram is central to the representation of a use case, but not enough
- Needs to come with a document stating:
  - Main actor
  - Secondary actors (optional)
  - Which system
  - Level of the use case (primary objective for the main actor, or sub-function?)
  - Glossary

  - Assumptions (which are assumed true for the correct execution)
  - Alternative use cases
  - Extensions of the use case

  - And the usual information (Name, date, version, …)

UML to model requirements

Conclusion

- Use cases allows:
  - To collect functional requirements
  - To analyze functional requirements
  - To discuss functional requirements
- It allows to understand the boundaries of the system
- It can be used to design the interfaces of the system
- It allows to validate requirements
- It can be part of the documentation

- WARNING: it is not a temporal diagram… Next week!