Requirements & Use Case Diagrams

DUM 04

SW requirements

- Requirement can be briefly defined as a „specification of what should be implemented“.
- Functional reqs define what behavior or what services will the system provide
- Non-functional reqs define attributes or restrictive conditions of given system (quality, technology, documentation, project management, etc.)
- Requirement defines „WHAT“, not „HOW“

Non-functional requirements

- Performance
- Capacity
- Availability
- Standards compliance
- Security
- Documentation
- Project organization
- Technology
- …

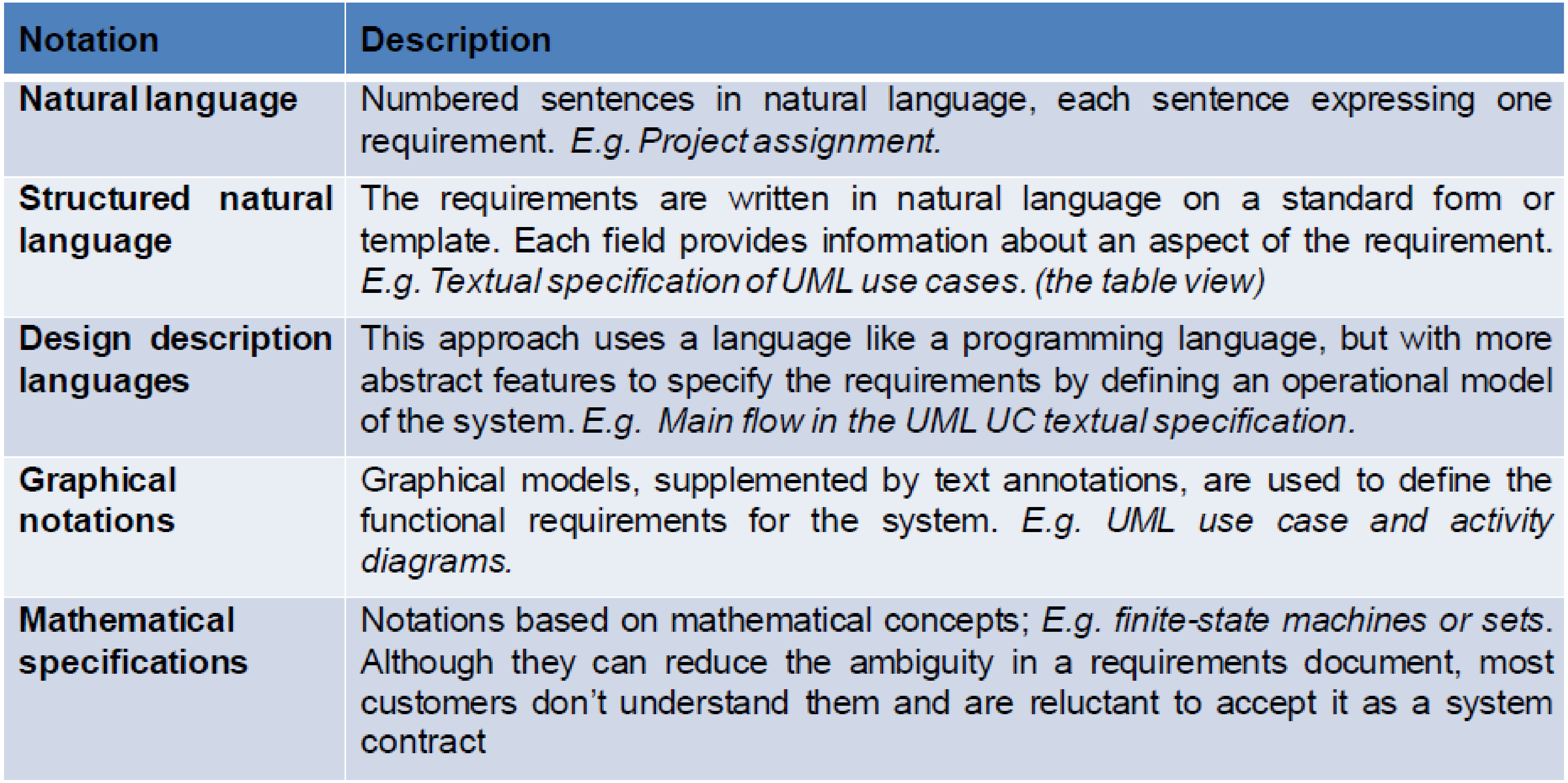
Non-functional requirements

Non-functional requirements are to be recorded and continually implemented during the project delivery. A catalogue record with following attributes can be used for this purpose:

- Identification number
- Last change date
- Name of author and recipient
- Requirements prioirity
- Brief description
- Detailed description
- Connection to other reqs
- Requirement complexity/complicacy

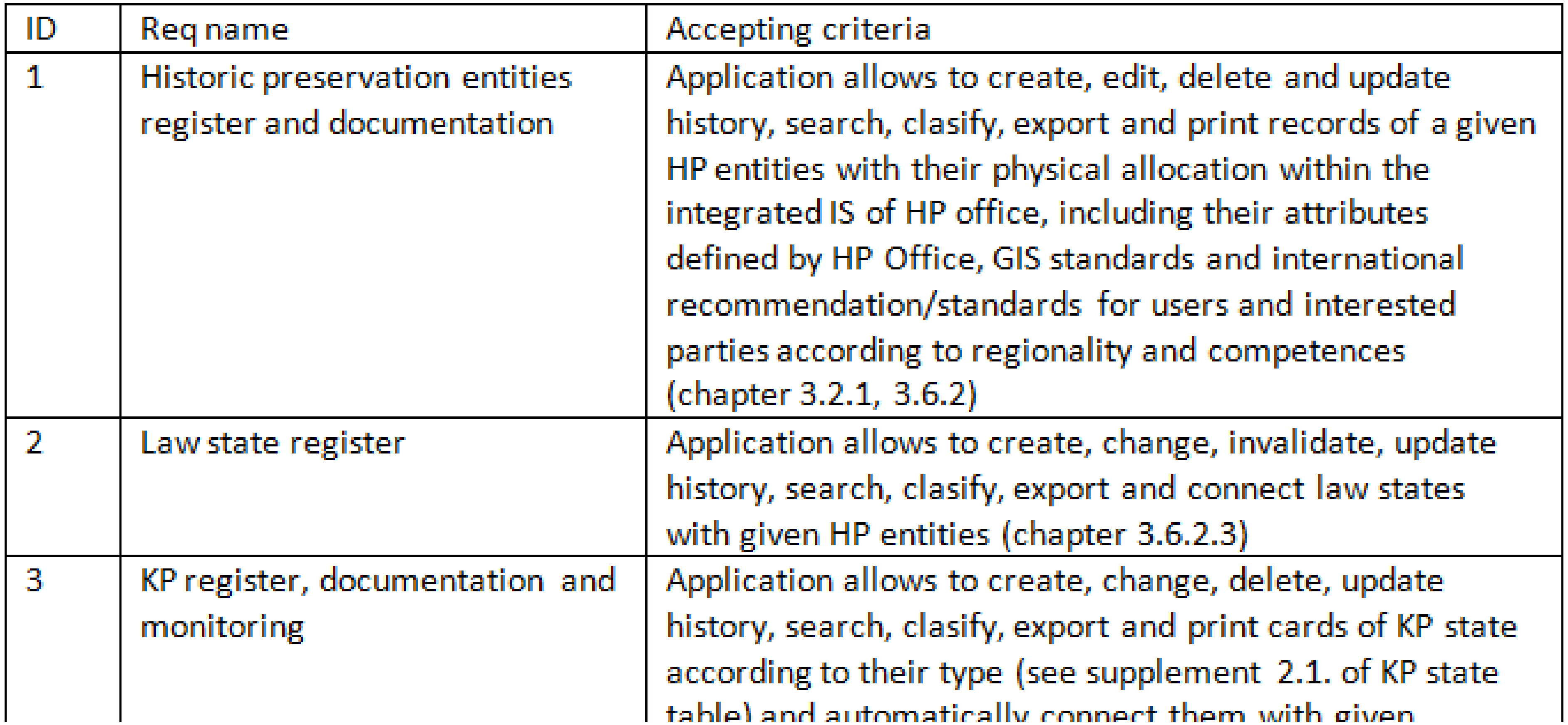
Functional reqs can be recorded in a „similar“ manner

Requirements record



Notation	Description
Natural language	Numbered sentences in natural language, each sentence expressing one requirement. *E.g. Project assignment.*
Structured natural language	The requirements are written in natural language on a standard fom or template. Each field provides infomation about an aspect of the requirement. *E.g. Textual specification of UML use cases. (the table view)*
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. *E.g. Main flow in the UML UC textual specification.*
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. *E.g. UML use case and activity diagrams.*
Mathematical specifications	Notations based on mathematical concepts; *E.g. finite-state machines or sets.* Although they can reduce the ambiguity in a requirements document, most customers don't understand them and are reluctant to accept it as a system contract

Example of requirements record



ID	Req name	Accepting criteria
1	Historic preservation entities register and documentation	Application allows to create, edit, delete and update history, search, clasify, export and print records of a given HP entities with their physical allocation within the integrated IS of HP office, including their attributes defined by HP Office, GIS standards and international recommendation/standards for users and interested parties according to regionality and competences (chapter 3.2.1, 3.6.2)
2	Law state register	Application allows to create, change, invalidate, update history, search, clasify, export and connect law states with given HP entities (chapter 3.6.2.3)
3	KP register, documentation and monitoring	Application allows to create, change, delete, update history, search, clasify, export and print cards of KP state according to their type (see supplement 2.1. of KP state table) and automatically connect them with given

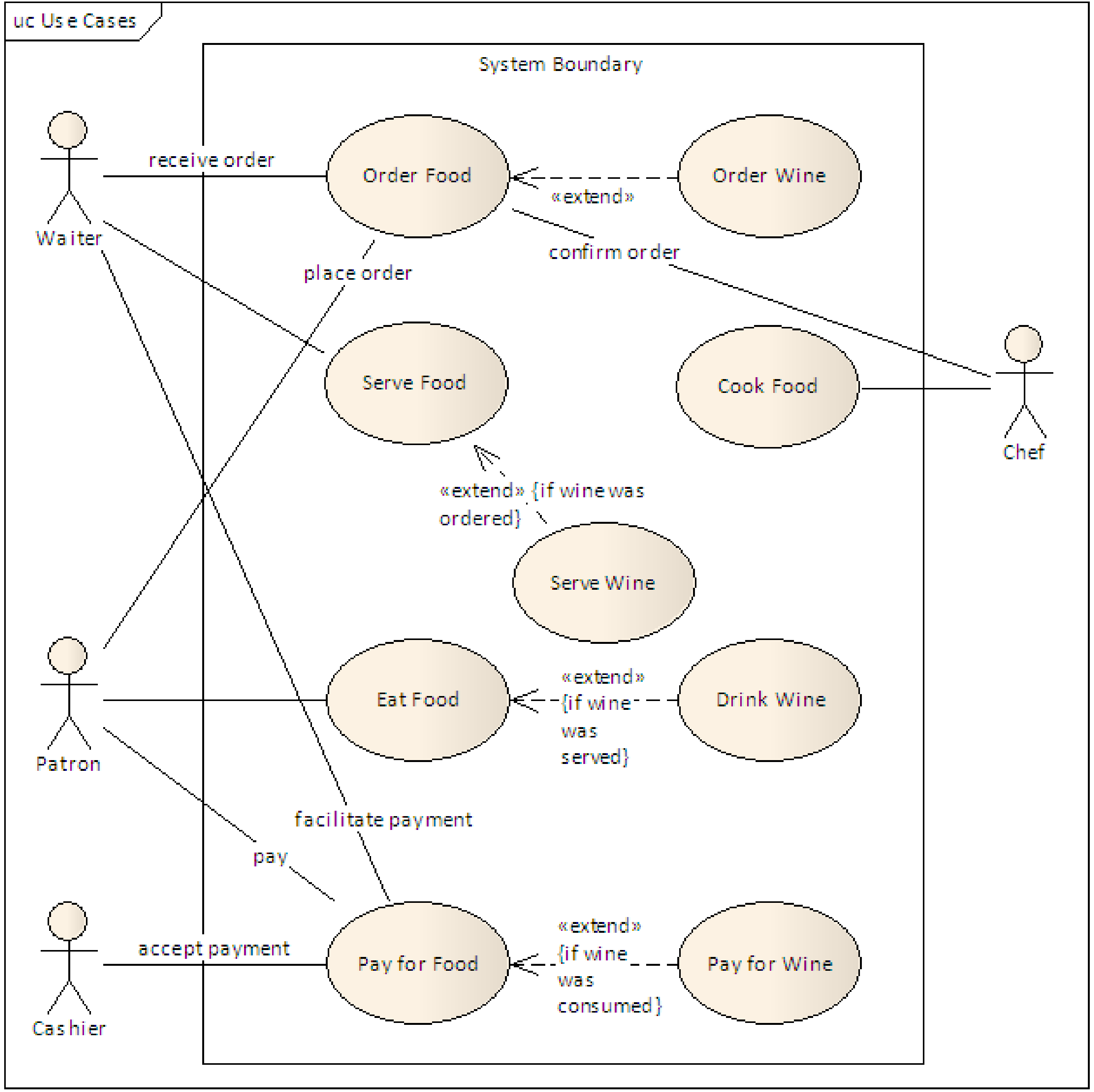
Functional requirements – use cases

- Use Case Diagram provides an outside view on the modelled system and therefore helps to define the borders of the system.

- It is a sequence of related transactions between user (usually a user in certain role but also a different system) and the system itself during a mutual dialogue.

- Main purpose is to record actors communicating with system and relations between services and their recipients through a visual and textual form that is comprehensive for both developers and customers (i.e. people who are going to use the system)

Use Case Diagram



Components and relationships

Use case (depicted as oval) – sequence of actions connected with an actor, can contain relationships

- Include – use case may contain other (Including – e.g. Edit text → Write text, Create table, etc.)
- Extend – use case may extend other (e.g. Open document → Import from other format, etc.)
- Generalisation (inheritance) – use case can be a special case of other use case

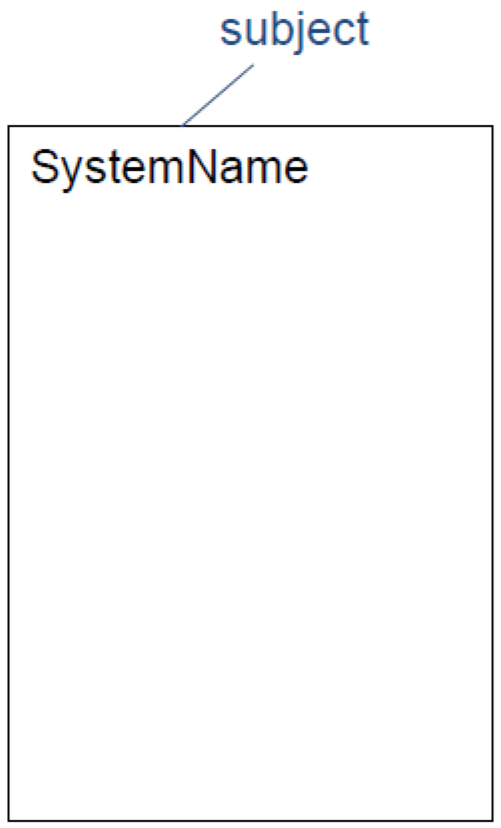
Actor/Participant (depicted as figure) – description of external objects participating in the process, may contain relationships

Generalisation (inheritance) – actor may be a special case of other actor

Components and relationships

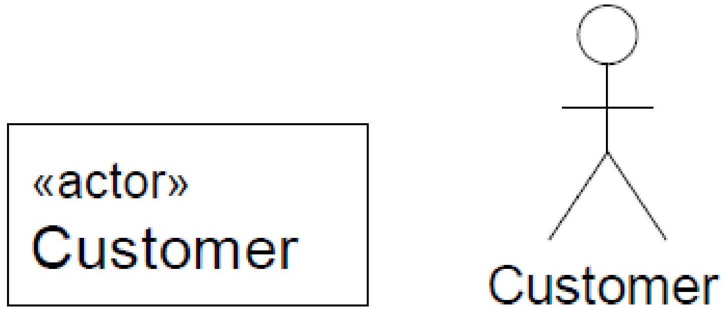
- We create a Use Case model containing:
  - **Subject** – the edge of the system
    - also known as the system boundary
  - **Actors** – who or what uses the system
  - **Use Cases** – things actors do with the system; functions the system should offer to its users
  - **Relationships** – between actors and use cases

- Can there be a direct relationship between actors?



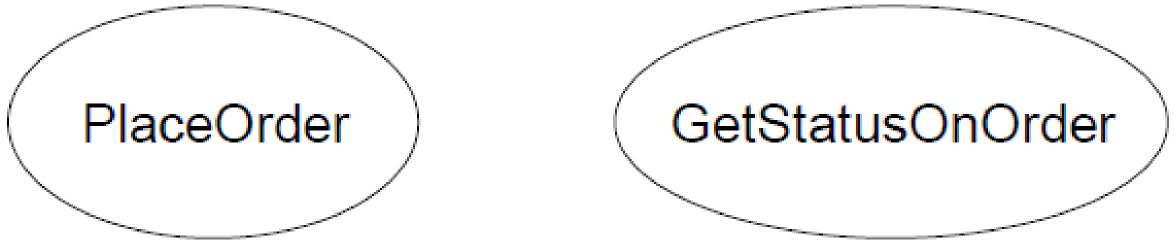
Components and relationships

- ✧ An actor is anything that interacts **directly** with the system
  - Actors identify who or what uses the system and so indicate where the system boundary lies
- ✧ Actors are **external** to the system
- ✧ An Actor specifies a **role** that some external entity adopts when interacting with the system
  - Can one actor represent two physical persons?
  - Can one physical person match to two actors?
  - Can there be two actors with the same name in the model?

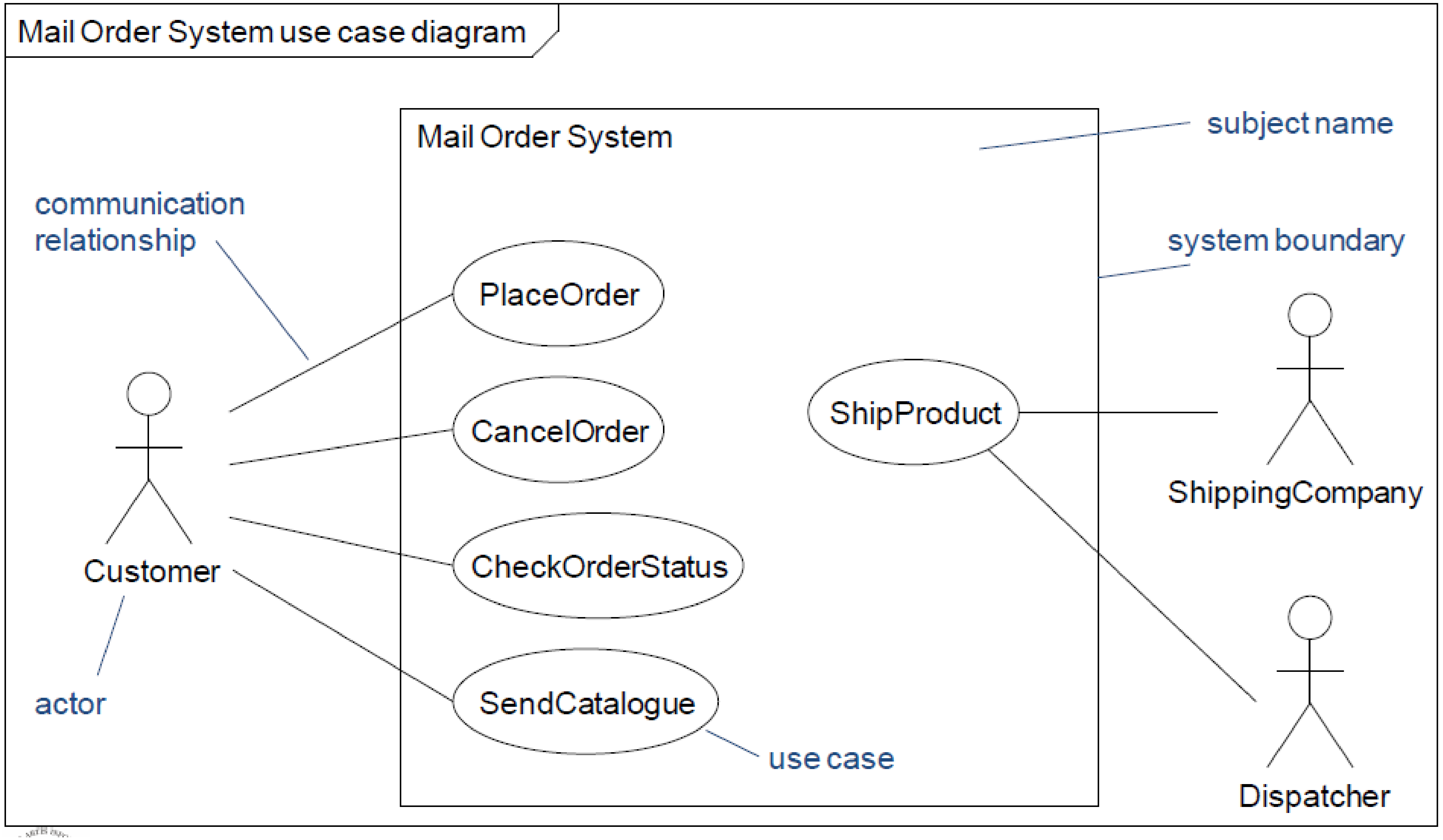


Components and relationships

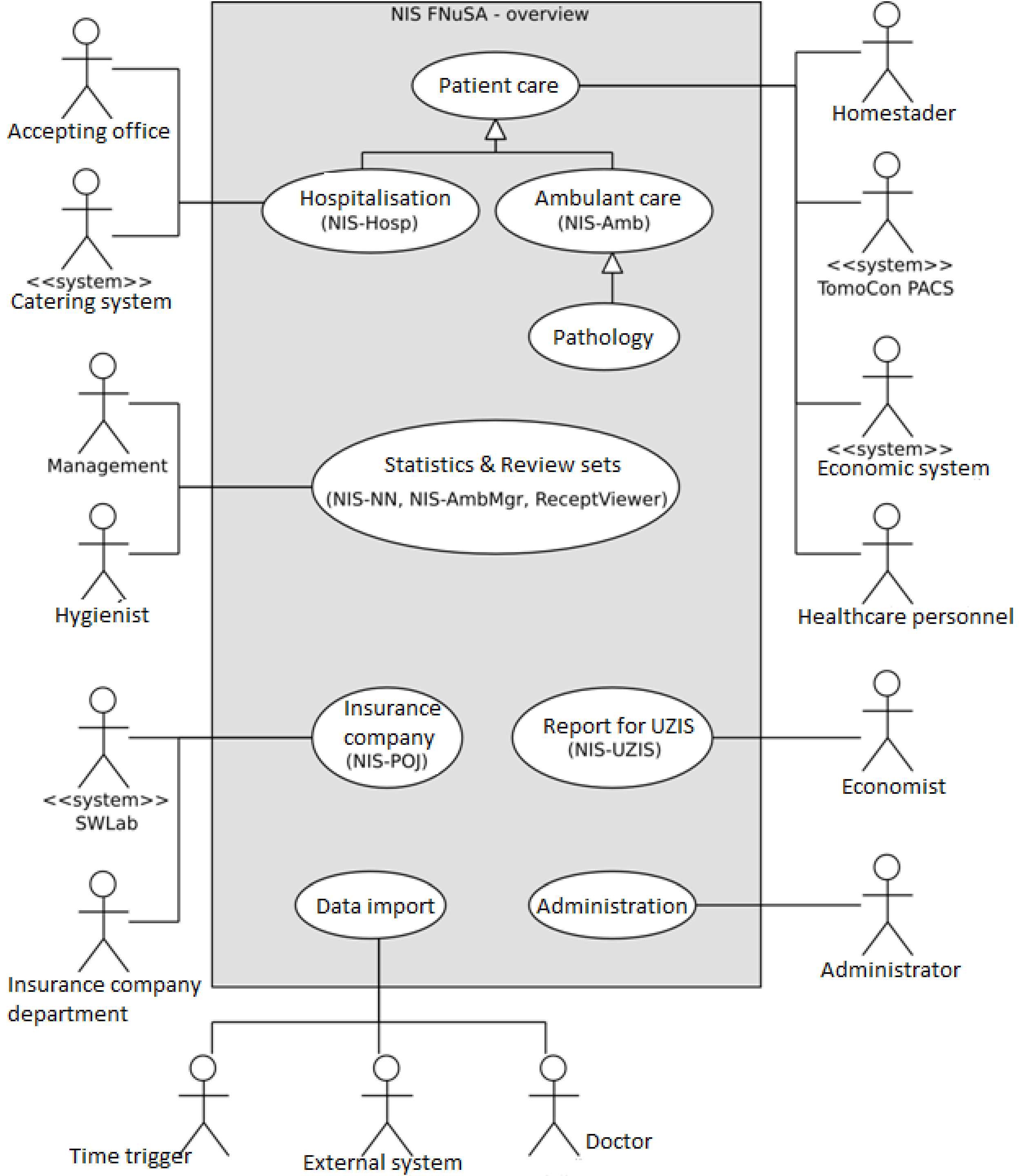
- A use case is something an actor needs the system to do. It is a “case of use” of the system by a specific actor.
- Use cases are always started by an actor
  - The **primary actor** triggers the use case
  - Zero or more **secondary actors** interact with the use case in some way
  - Does the UC diagram tell me which actor is primary/secondary?
- Use cases are always written from the **point of view of the actors**.

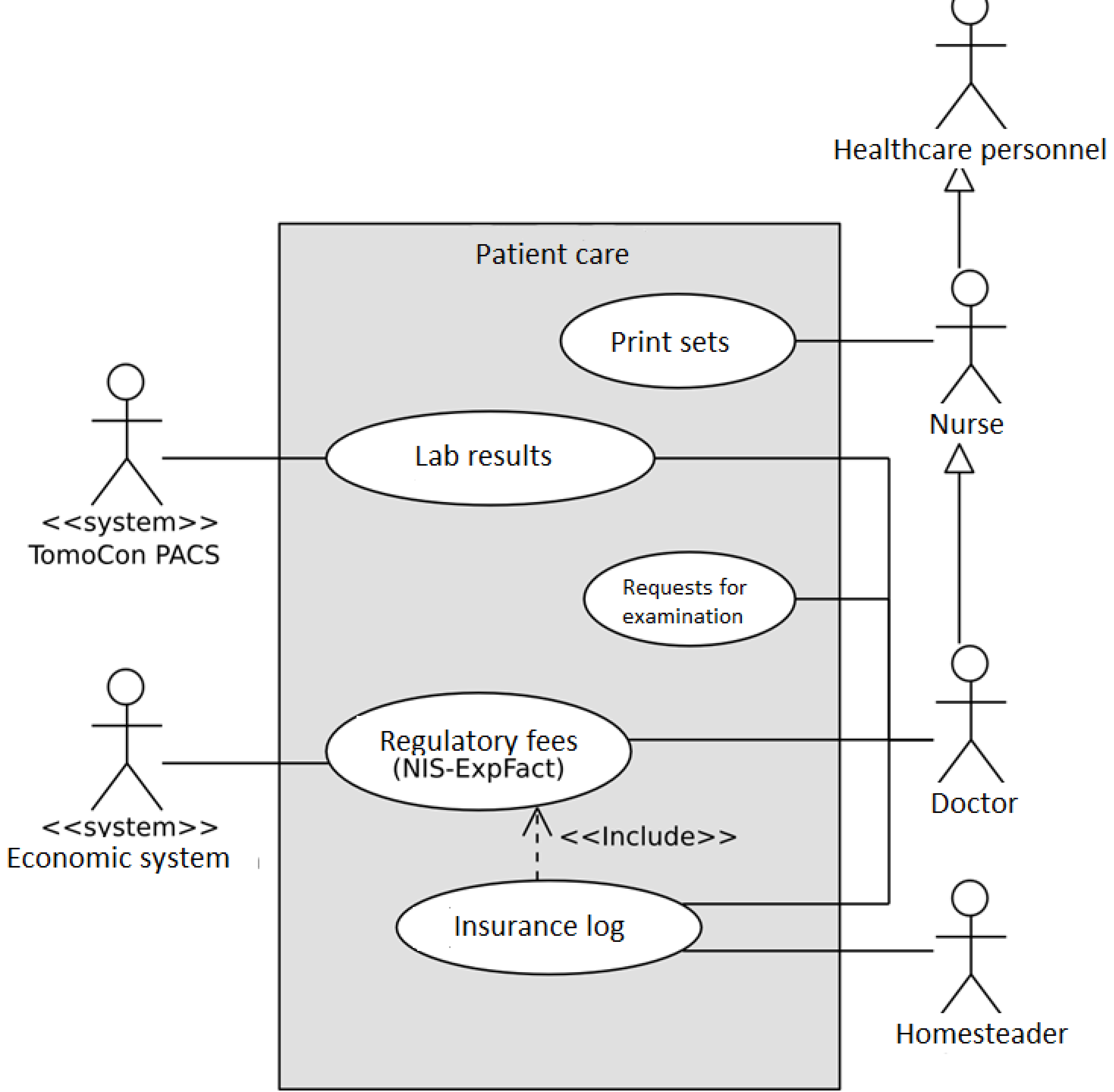


Components and relationships



use case name	Use case: PaySalesTax
use case identifier	ID: 1
brief description	Brief description: Pay Sales Tax to the Tax Authority at the end of the business quarter.
the actors involved in the use case	Primary actors: Time
	Secondary actors: TaxAuthority
the system state before the use case can begin	Preconditions: 1. It is the end of the business quarter.
the actual steps of the use case	Main flow: implicit time actor 1. The use case starts when it is the end of the business quarter. 2. The system determines the amount of Sales Tax owed to the Tax Authority. 3. The system sends an electronic payment to the Tax Authority.
the system state when the use case has finished	Postconditions: 1. The Tax Authority receives the correct amount of Sales Tax.
alternative flows	Alternative flows: None.


NIS FNuSA - overview
Patient care
Hospitalisation (NIS-Hosp)
Ambulant care (NIS-Amb)
Pathology
Statistics & Review sets (NIS-NN, NIS-AmbMgr, ReceptViewer)
Insurance company (NIS-POJ)
Report for UZIS (NIS-UZIS)
Data import
Administration
Accepting office
<<system>> Catering system
Management
Hygienist
<<system>> SWLab
Insurance company department
Homestader
<<system>> TomoCon PACS
<<system>> Economic system
Healthcare personnel
Economist
Administrator
Time trigger
External system
Doctor


Healthcare personnel
Patient care
Print sets
Nurse
Lab results
<<system>>
TomoCon PACS
Requests for examination
Regulatory fees (NIS-ExpFact)
Doctor
<<system>>
Economic system
<<Include>>
Insurance log
Homesteader