UML use case diagram & template

Tamás Ambrus, István Gansperger

Eötvös Loránd University
ambrus.thomas@gmail.com

UML use case diagram

UML

Unified Modeling Language has a wide range of support of diagrams that help to formalize and understand specification requirements.

Last time we learned about class diagrams one can illustrate the stucture of the system with. This type of diagram belongs to the **structural diagrams**' group.

Let's learn a new diagram type from the **behavioral diagrams**' group: use case diagram.

Use case diagram

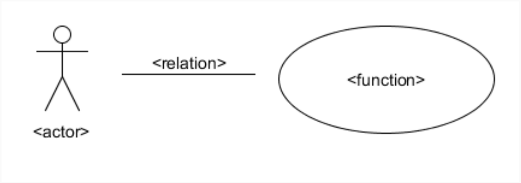
Use case diagrams illustrate how the users cooperate with the system

- it is the standardized way of describing the functional requirements,
- it is based on clarity, we use these diagrams with detailed descriptions usually.

Basic components

The elements of diagrams are:

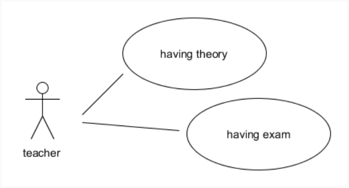
- actor: the person who uses the system,
- function: one of the services of the system,
- relation: relationship between the actor and a function, or function and function (or even between actors).
- group: some functions can be collected into a group



Basic components

Types of relations:

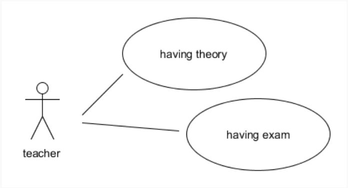
- usage: actor has resort to a function of the system (can only be drawn between actors and functions)



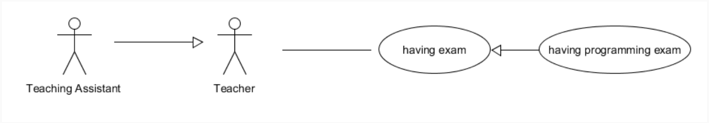
Basic components

Types of relations:

- usage: actor has resort to a function of the system (can only be drawn between actors and functions)



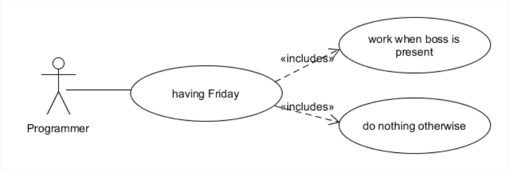
- generalization: a function or an actor is more specific than the other one



Basic components

Types of relations:

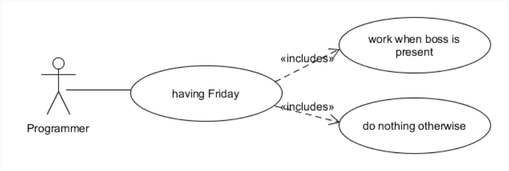
- includes: a function contains another function, so *having a Friday may contain other steps too, but what we know is that it will contain these two steps assuredly*



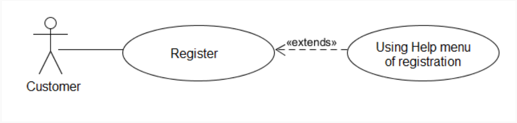
Basic components

Types of relations:

- includes: a function contains another function, so *having a Friday may contain other steps too, but what we know is that it will contain these two steps assuredly*



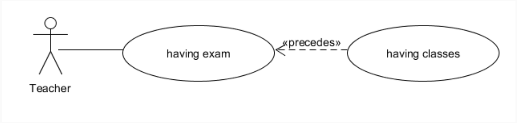
- extends: a function may extend another one, so *the customer can use the help menu on the registration page*



Basic components

Types of relations:

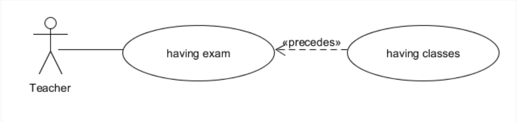
- precedes: a function must happen before another one, so *the teacher cannot hold an exam as long as there are no classes*



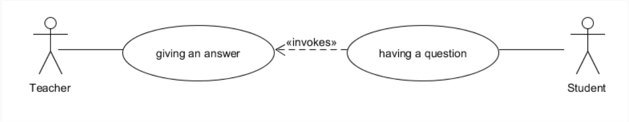
Basic components

Types of relations:

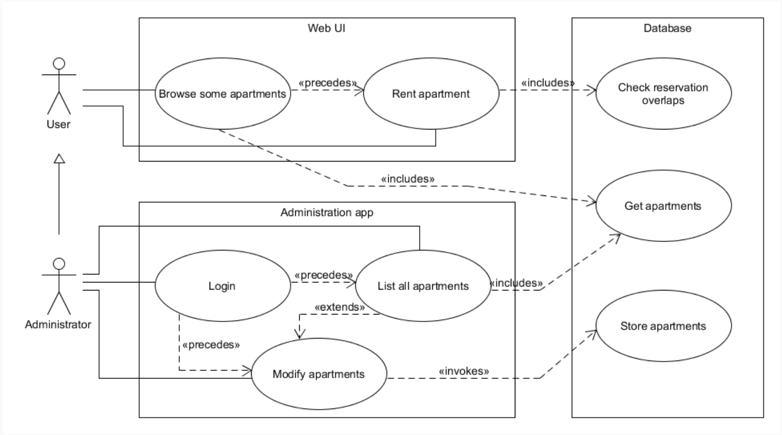
- precedes: a function must happen before another one, so *the teacher cannot hold an exam as long as there are no classes*



- invokes: a function will be followed by another one automatically, so *any questions coming from the students will cause the teacher to answer, there's nothing to do against it*

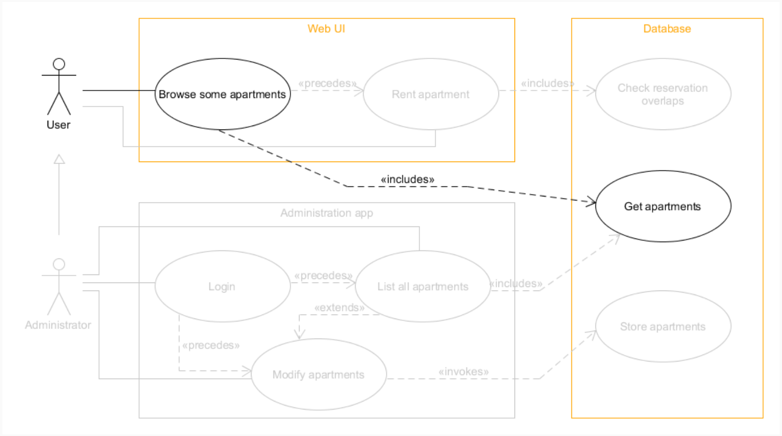


Use case diagram example



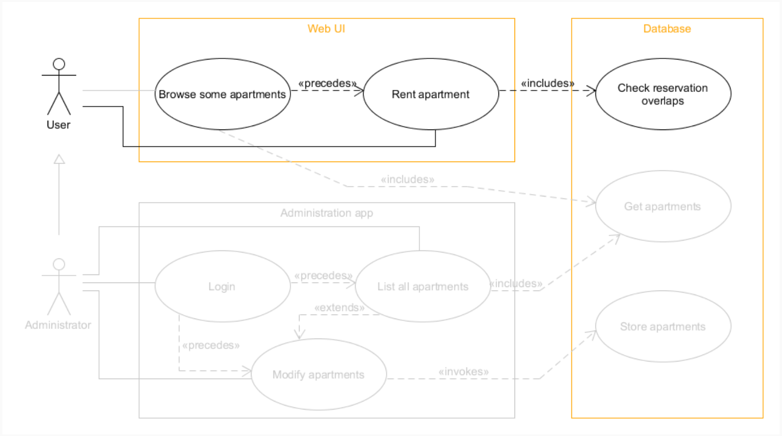
Let's look a simple apartment renting example step-by-step. How can we "read" a diagram like this?

Use case diagram example - use case 1



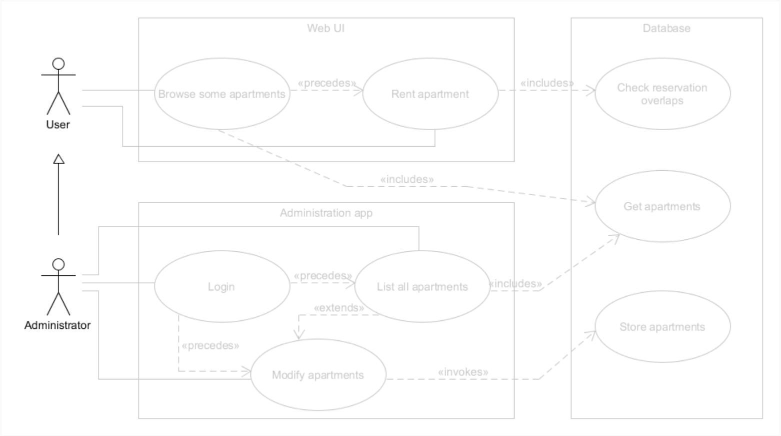
There are users who can browse the apartments on a web page. During browsing the database has to load the apartments.

Use case diagram example - use case 2



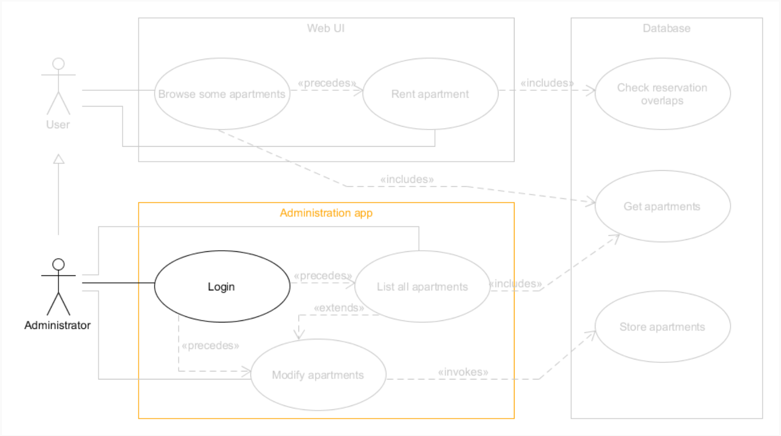
Users can rent apartments. Before doing so, they have to browse them. The renting process contains checks.

Use case diagram example - special user



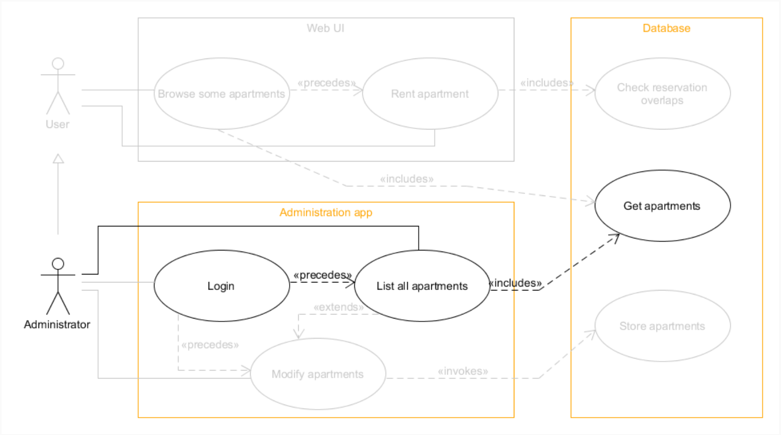
Administrators are special users, meaning they can do everything that users can (browse and rent), but they can do more...

Use case diagram example - use case 3



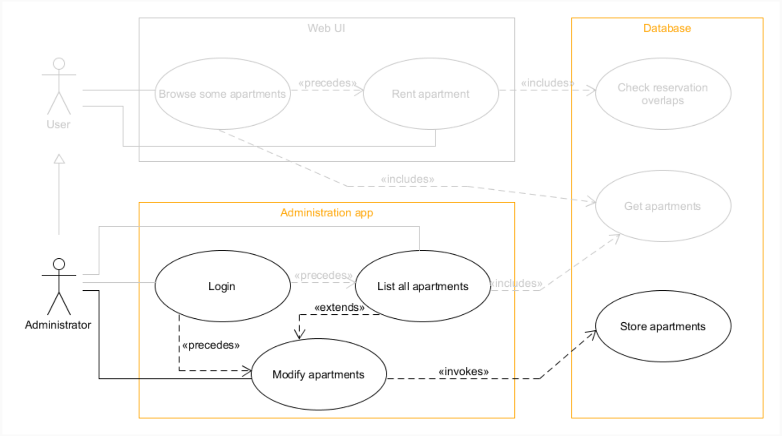
They can login into an application that is made only for administrators.

Use case diagram example - use case 4



They can list the apartments. They have to login before. Database has to select all as part of this process.

Use case diagram example - use case 5



They can modify apartments, but they have to login before. As a help, they can even list them if needed. Everything will be saved.

Cons

Thoughts can be in a whirl, many details remain hidden from the programmer after having a look at a use case diagram. One could ask e.g. how do the administrators modify the apartments? What can they edit? What exception could occur? This is an important part of the use case.

Or what kind of database are we going to use? This is an important question as well, but since it is not a use case, it isn't the right place to ask it.

Anyway, you are right if you think that UC diagram isn't detailed. We will use UC templates to gather details.

Pros

So what are these diagrams good for?

- summarizes use cases into a short, well-structured diagram,
- newcomers can understand requirements faster than learning templates for hours and hours,
- by drawing one, questions may come into your mind,
- much easier to draw this first than gathering low level details.

Use case template

Use case template

Use case templates give us a semi-formal way to write the requirements specification. It's basically a table with pre-defined fields that have to be filled in.
Each use case has a completely filled table.

The best is to use them in conjunction as the diagram is easier to read, but does not contain information about the systematic interaction between a use case and an actor.

UC template fields

These are the fields that have to be filled:

- Use case name,
- Actors,
- When,
- Main-scenario,
- Exception scenarios,
- Pre-conditions,
- Post-conditions.