Politecnico
di Milano

How to reason about design? An example

The taxi service case

Exercise

- Assume that the Milano municipality asks you to develop a digital taxi service
- The municipality wants to replace the taxi stops available on the streets with the online service
- The Major sends you a written description of what they would like to have (see next two slides)
- Define the architecture of the system

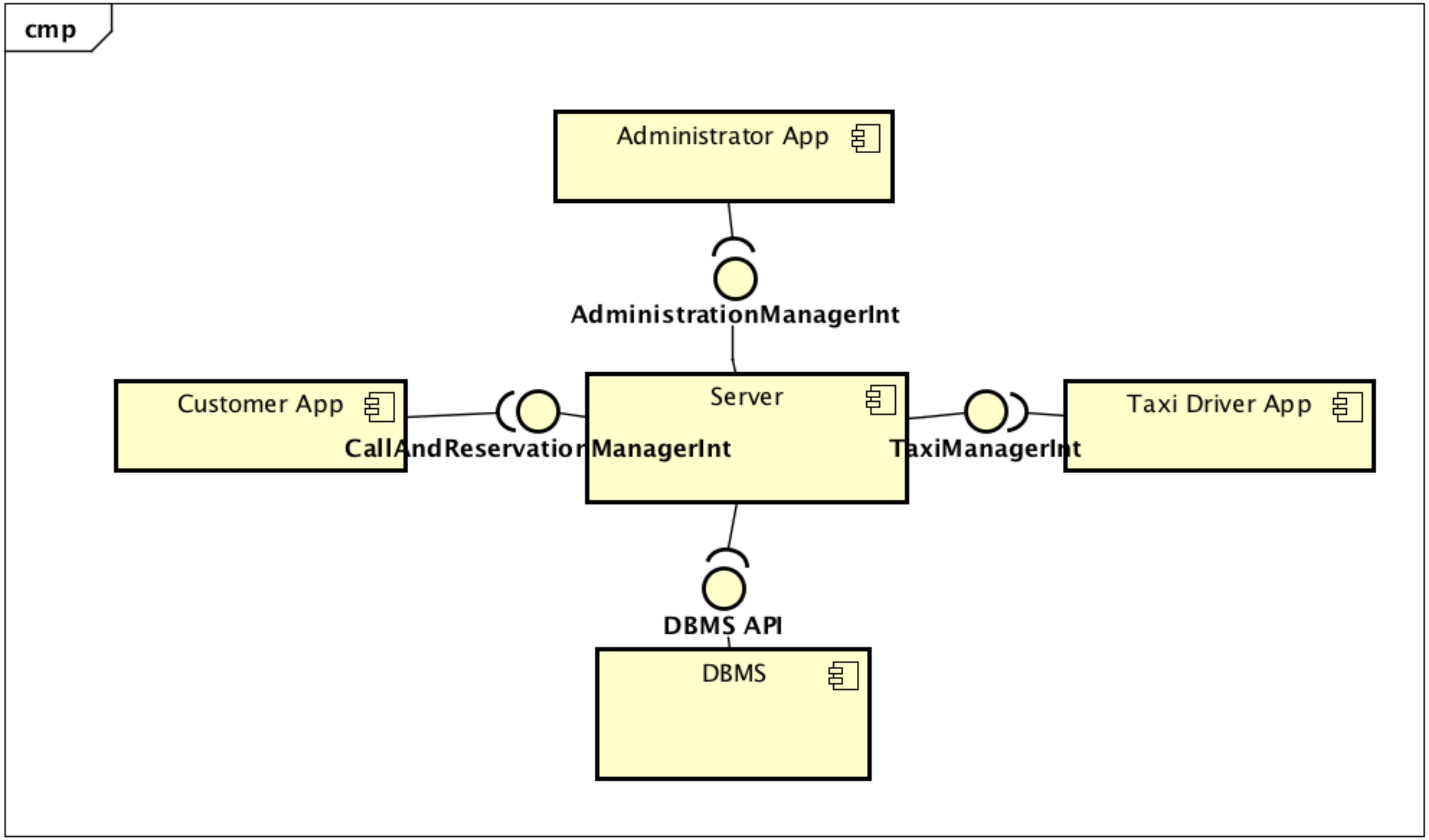
MyTaxi Service (1)

- The government of a large city aims at optimizing its taxi service. In particular, it wants to:
  - i) simplify the access of passengers to the service, and
  - ii) guarantee a fair management of taxi queues.
- Passengers can request a taxi either through a web application or a mobile app. The system answers to the request by informing the passenger about the code of the incoming taxi and the waiting time.
- Taxi drivers use a mobile application to inform the system about their availability and to confirm that they are going to take care of a certain call.
- The system guarantees a fair management of taxi queues. In particular, the city is divided in taxi zones (approximately 2 km2 each). Each zone is associated to a queue of taxis. The system automatically computes the distribution of taxis in the various zones based on the GPS information it receives from each taxi. When a taxi is available, its identifier is stored in the queue of taxis in the corresponding zone.

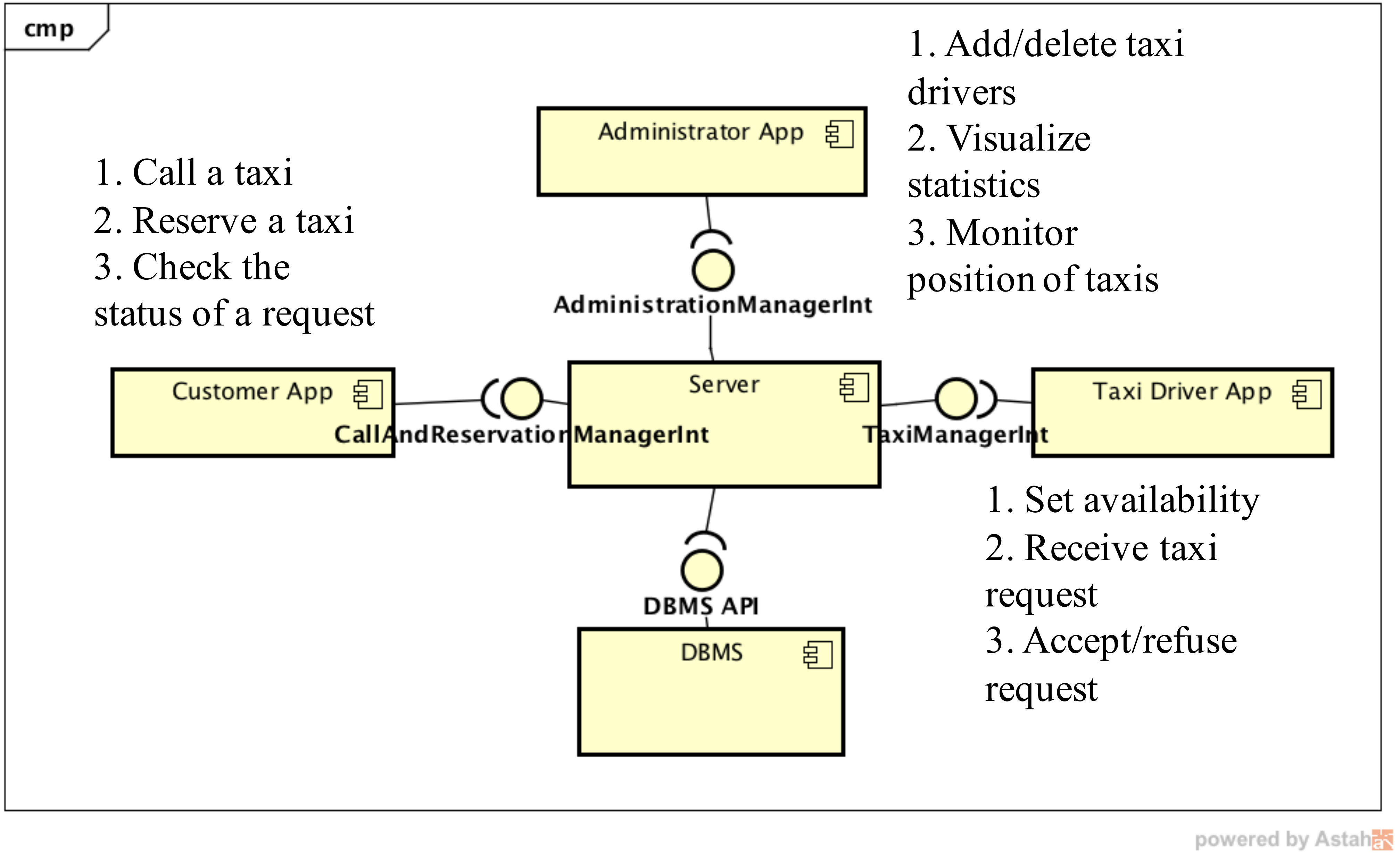
MyTaxi Service (2)

- When a request arrives from a certain zone, the system forwards it to the first taxi queuing in that zone. If the taxi confirms, then the system will send a confirmation to the passenger. If not, then the system will forward the request to the second in the queue and will, at the same time, move the first taxi in the last position in the queue.
- A user can also reserve a taxi by specifying the origin and the destination of the ride. The reservation has to occur at least two hours before the ride. In this case, the system confirms the reservation to the user and allocates a taxi to the request 10 minutes before the meeting time with the user.

Main components



Allocation of functionalities to the external components

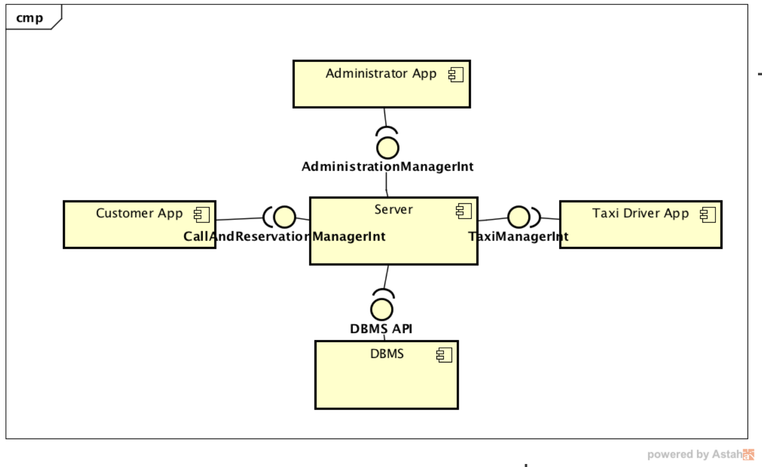


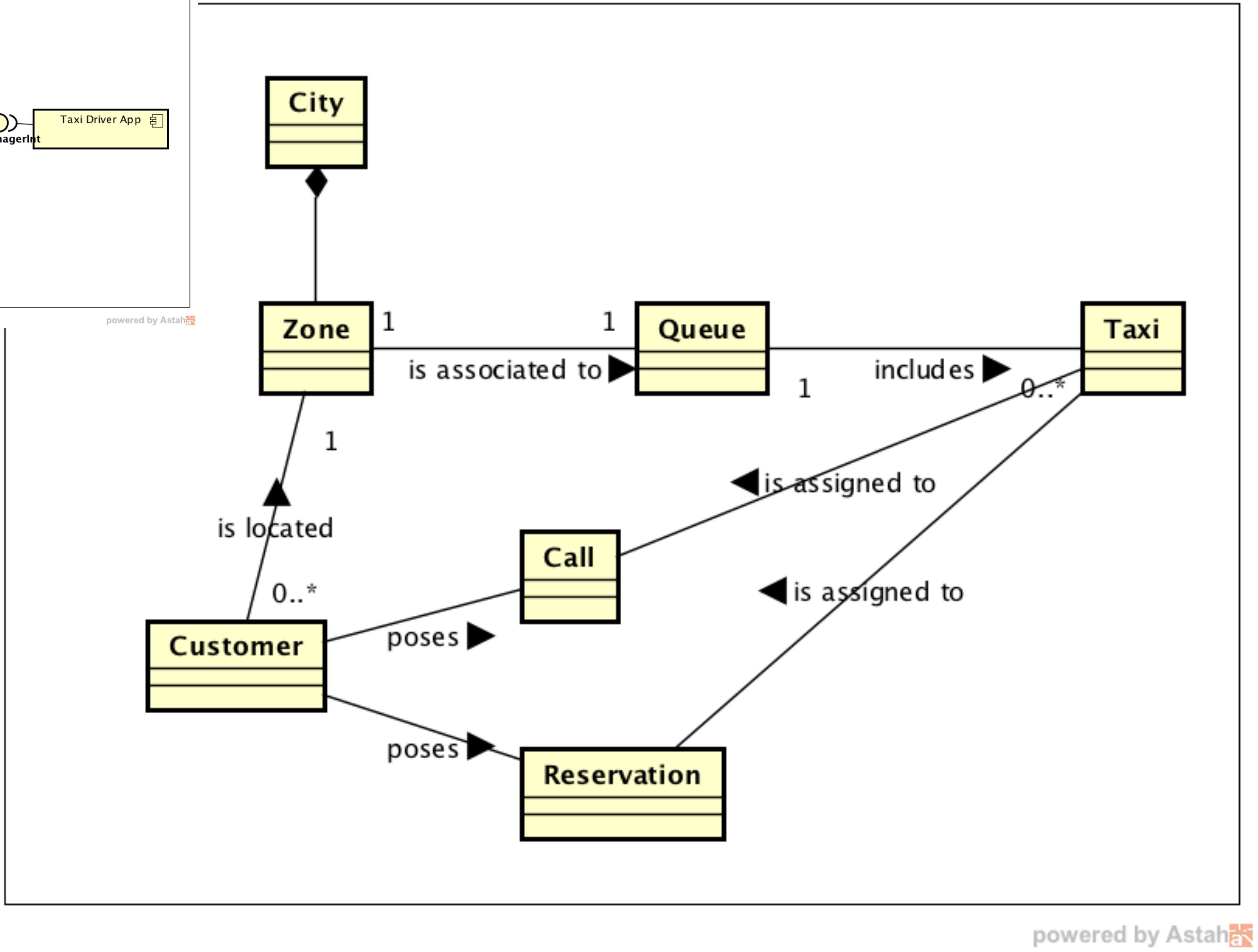
1. Call a taxi
2. Reserve a taxi
3. Check the status of a request

1. Add/delete taxi drivers
2. Visualize statistics
3. Monitor position of taxis

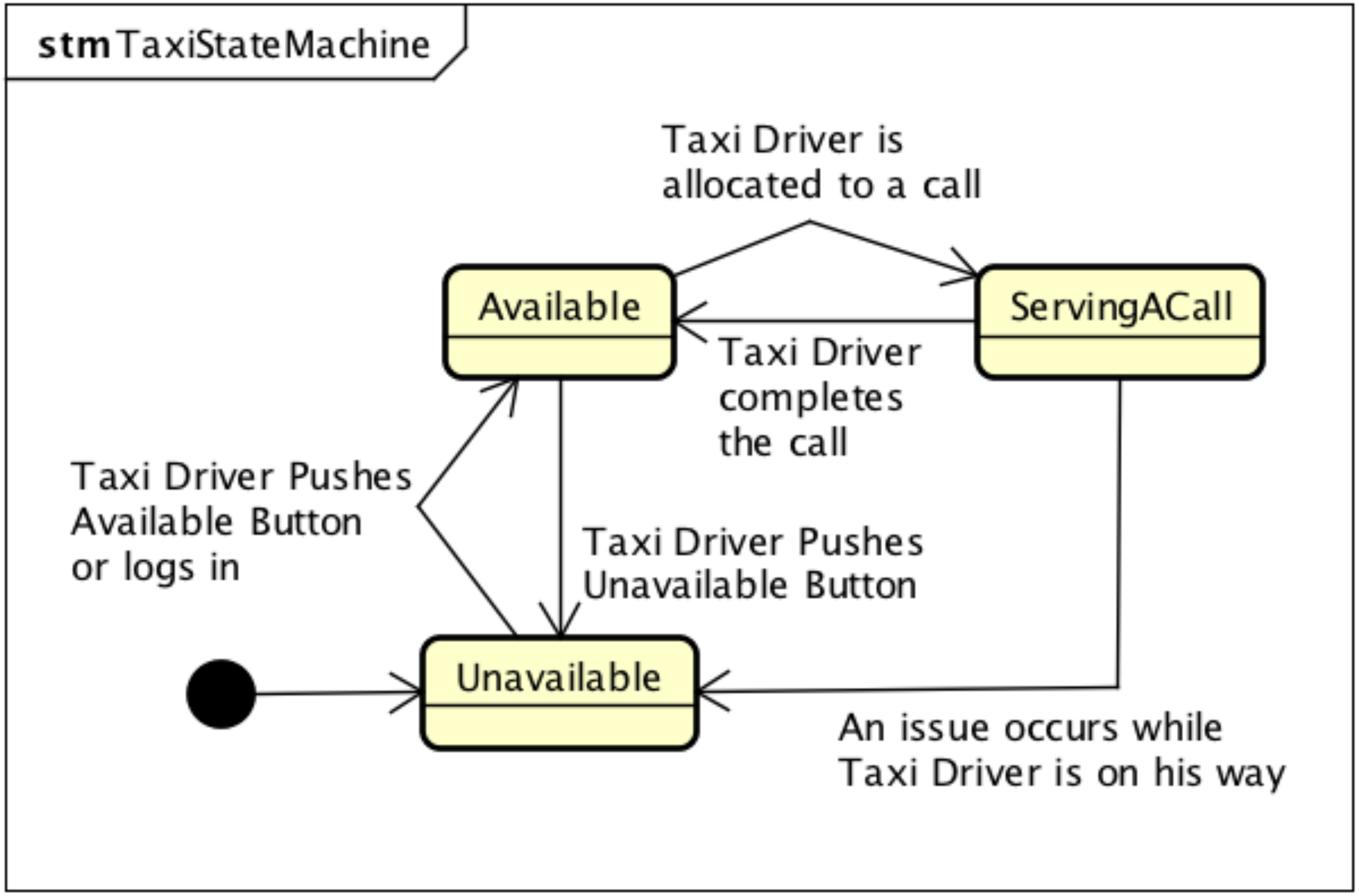
1. Set availability
2. Receive taxi request
3. Accept/refuse request

Server: objects wrapping data structures





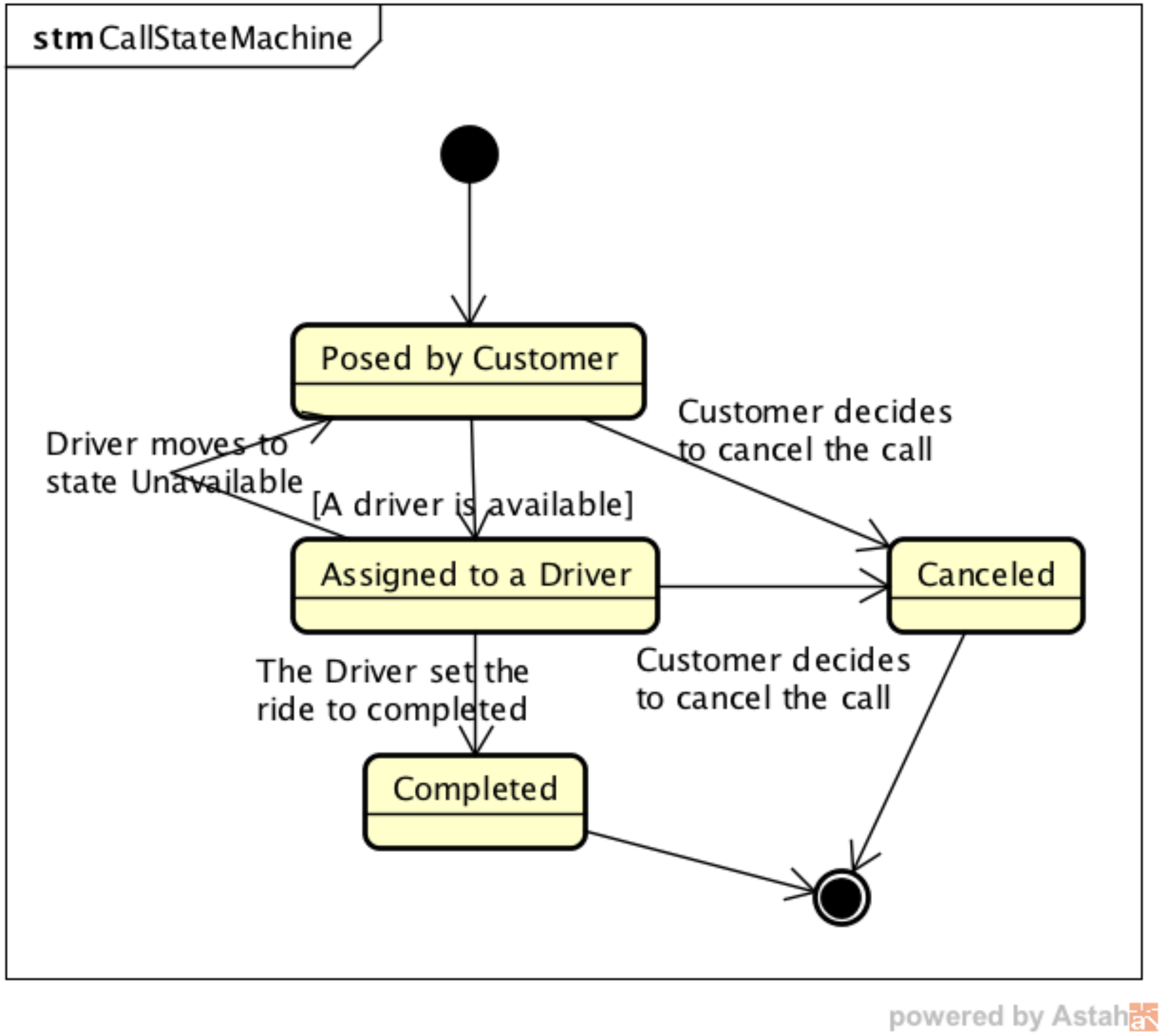
StateCharts for relevant data structures



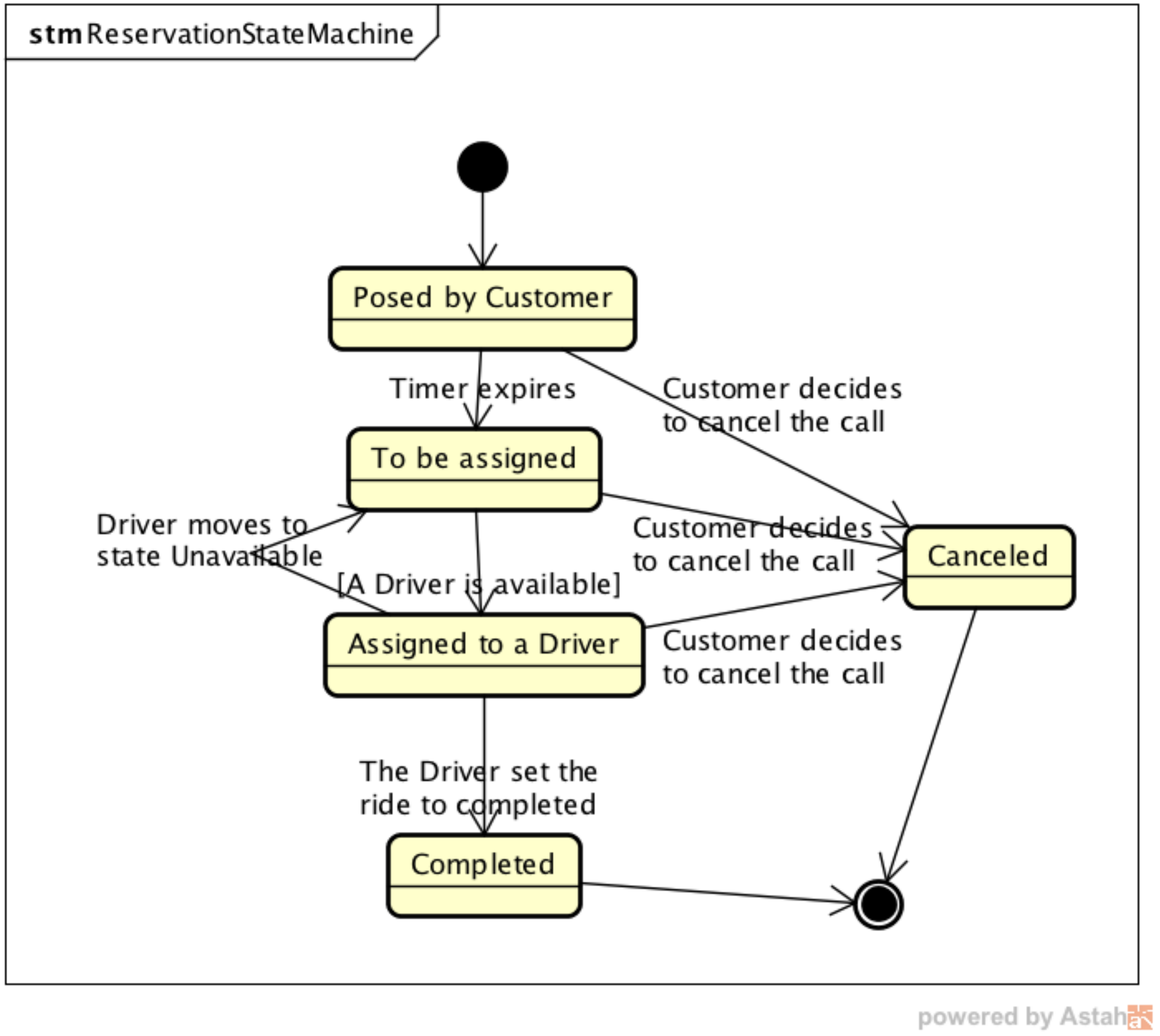
stm TaxiStateMachine
Taxi Driver is allocated to a call
Available
ServingACall
Taxi Driver completes the call
Taxi Driver Pushes Available Button or logs in
Taxi Driver Pushes Unavailable Button
Unavailable
An issue occurs while Taxi Driver is on his way

powered by Astah

StateCharts for relevant data structures


stm CallStateMachine
Posed by Customer
Driver moves to state Unavailable
[A driver is available]
Customer decides to cancel the call
Assigned to a Driver
Canceled
The Driver set the ride to completed
Customer decides to cancel the call
Completed
powered by Astah

StateCharts for relevant data structures

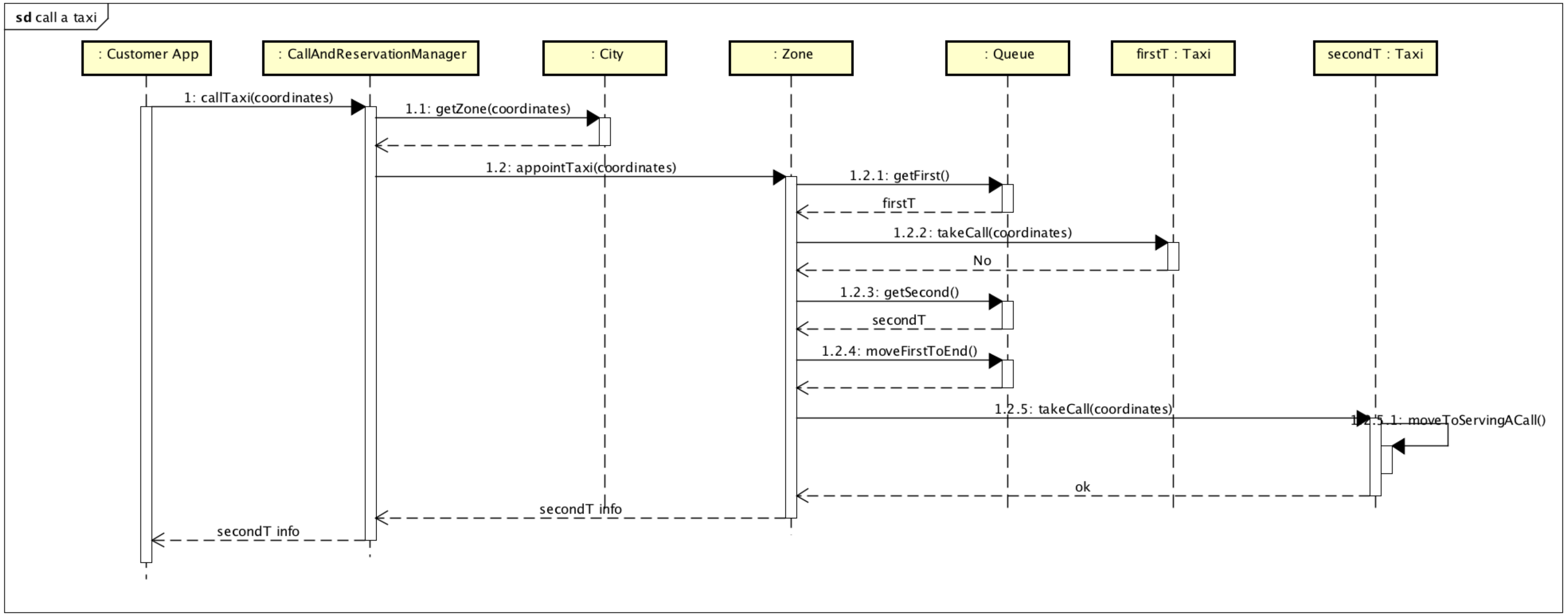

stm ReservationStateMachine
Posed by Customer
Timer expires
Customer decides to cancel the call
To be assigned
Driver moves to state Unavailable
Customer decides to cancel the call
Canceled
[A Driver is available]
Assigned to a Driver
Customer decides to cancel the call
The Driver set the ride to completed
Completed
powered by Astah

Server: Façade components

- CallAndReservationManager
  - Interacts with the customer app to acquire calls and reservations for taxis and to check status of calls
  - It waits for requests from the customer app
  - Pushes info back to the customer app when the call/reservation is assigned to a taxi driver
- CustomerRegistrationManager
  - Interacts with the customer app to acquire data for customer registration
- LoginManager
  - It manages login of all users
- AdministrationManager
- TaxiManager

How do we understand how components are connected together?

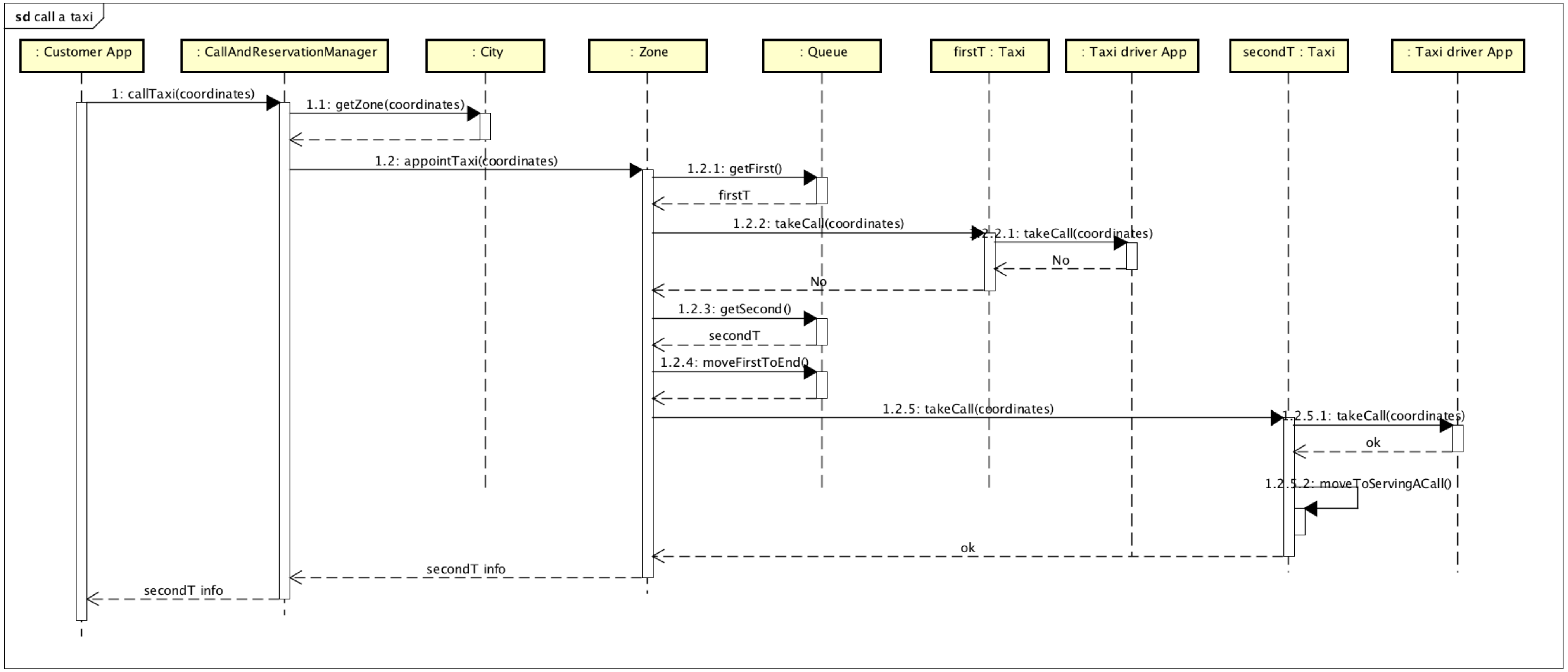
- Try with sequence diagrams
- Use them to identify all main operations to be offered by components
- … and the way components interact



powered by Astah

Question 1

- Can firstT and secondT make decisions on behalf of taxi drivers?
- Maybe not! Updated diagram:

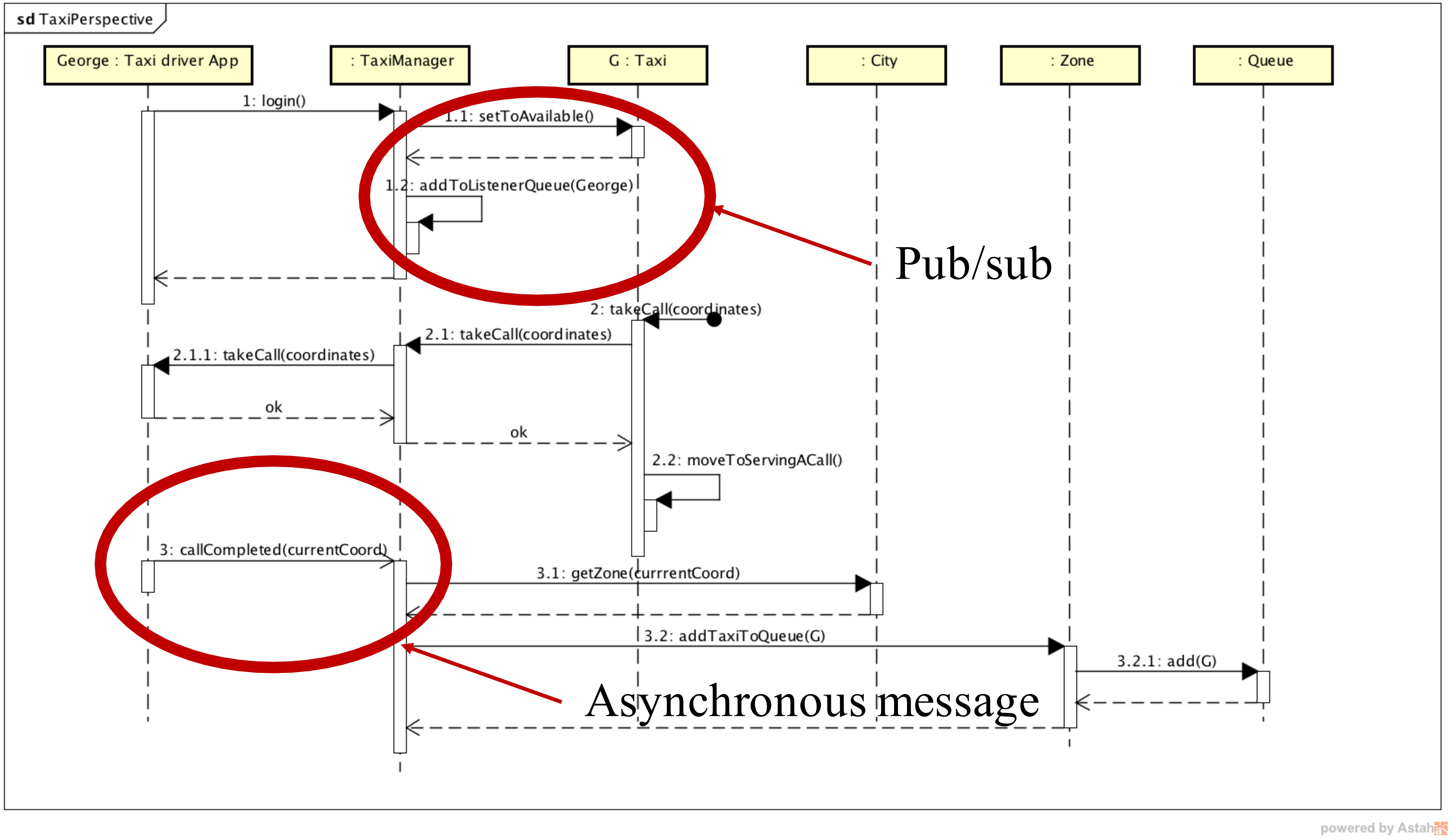


Question 2

- Can firstT and secondT interact directly with the Taxi driver App?
- Shouldn’t we delegate this interaction to the TaxiManager component?
- You do the change (but see next slide …)

We need to continue exploring behaviors…

- How can server-side components send messages to the Taxi driver App?



Other issues

- How many instances per components?
  - All managers could be available in multiple instances, one per request
  - They could be stateless
  - What about components encapsulating data structures?
- How to manage communication between distributed components?
- Which database?
- How to interface with database?
- Studying all this and assessing our technological ability, schedule, budget, should drive us to the selection of some underlying technology
  - Or, alternatively, to rely only on the standard OS and communication levels (see the NginX case)

So, how do we continue?

- Try to answer to all questions that come into your mind
- Continue exploring all possible behaviors
- Highlight all possible kinds of communication between distributed components
- You will be soon ready to define your architectural views and the interfaces of your components
- When do you stop?
  - Ideally, when you know you are at a point where you are able to start splitting the work and coding

Question

- Aren’t we missing some elements/components from our design level diagrams?
  - Yes, we do
  - The analysis is incomplete
  - If you check the class diagram against the sequence you will see that we are missing some interactions…
  - We encourage you to further work on this

Algorithms

- What about algorithms?
  - Where could we find complex algorithms in this example?
    - If we want to balance queues
    - If we want to look for available taxis in nearby queues when needed
    - If we want to handle taxi sharing
  - You will see that in many cases you can start from something pre-existing
  - Be aware that a bad algorithm results in bad performance!

A note on non functional requirements

- We know they have a strong impact on architecture
  - Some can be addressed with stateless components, replication, load balancing
- We have seen some architectural cloud patterns to address them
- Others you will see in other courses
- Keep both functional and non-functional requirements into account while reasoning on your architecture…