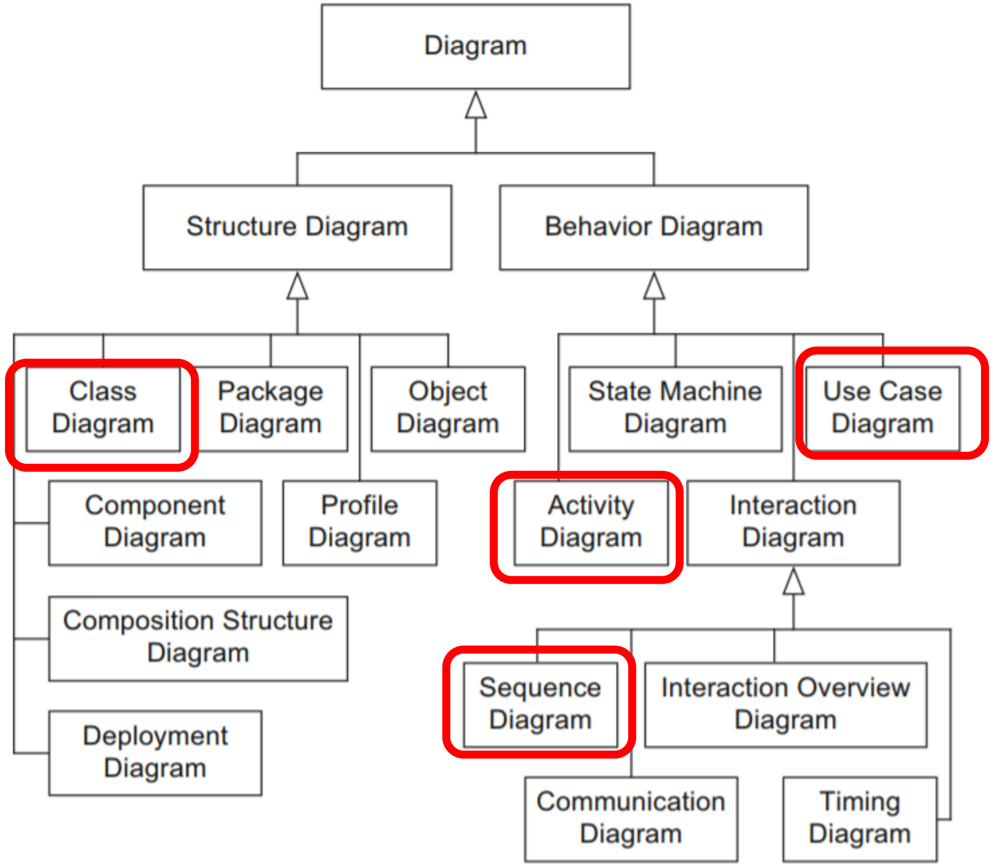
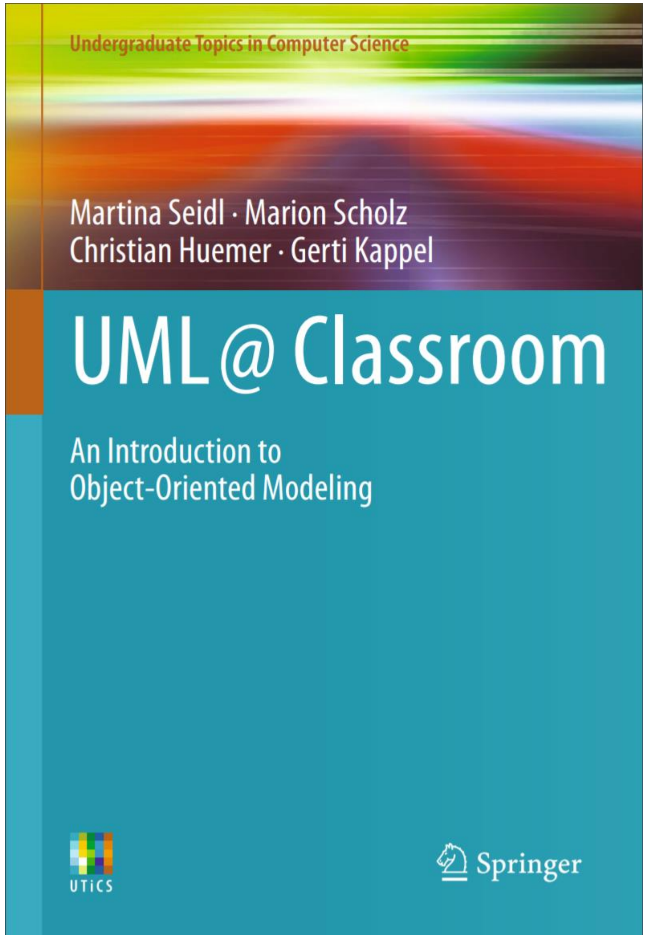
Lecture 6: Applications of UML

UML Diagrams



Reference for UML

- Freely available online
- Search from our library website



UML Drawing Tools

- Microsoft Visio can draw basic UML diagrams
  - Available from the library
- Visual Paradigm (Community edition)
  - https://www.visual-paradigm.com/download/community.jsp
- IBM Rational Rose
  - Cracked version online (not recommended)

Example: Information system for restaurants

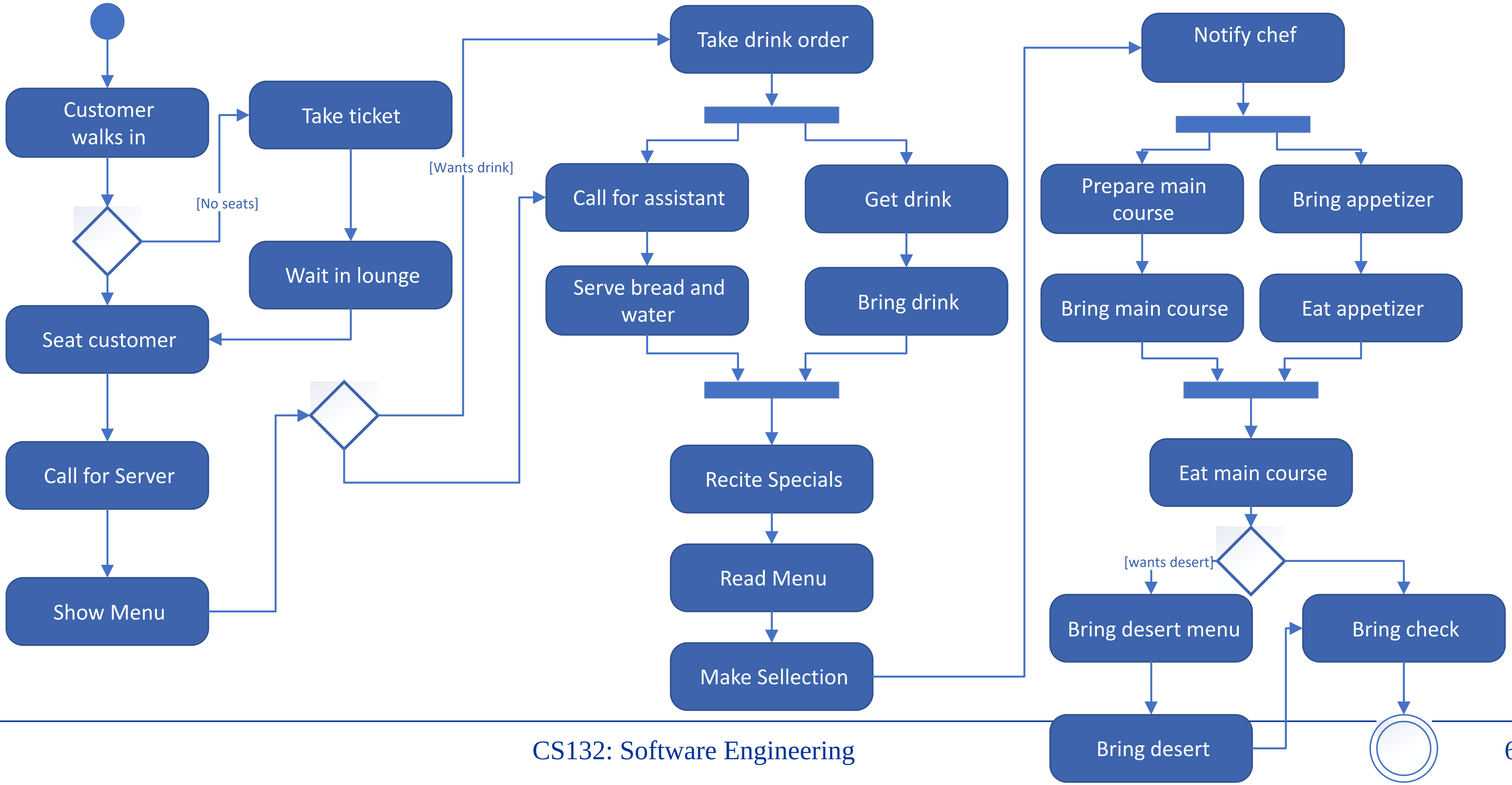
- The owner of restaurant A would like to improve service efficiency

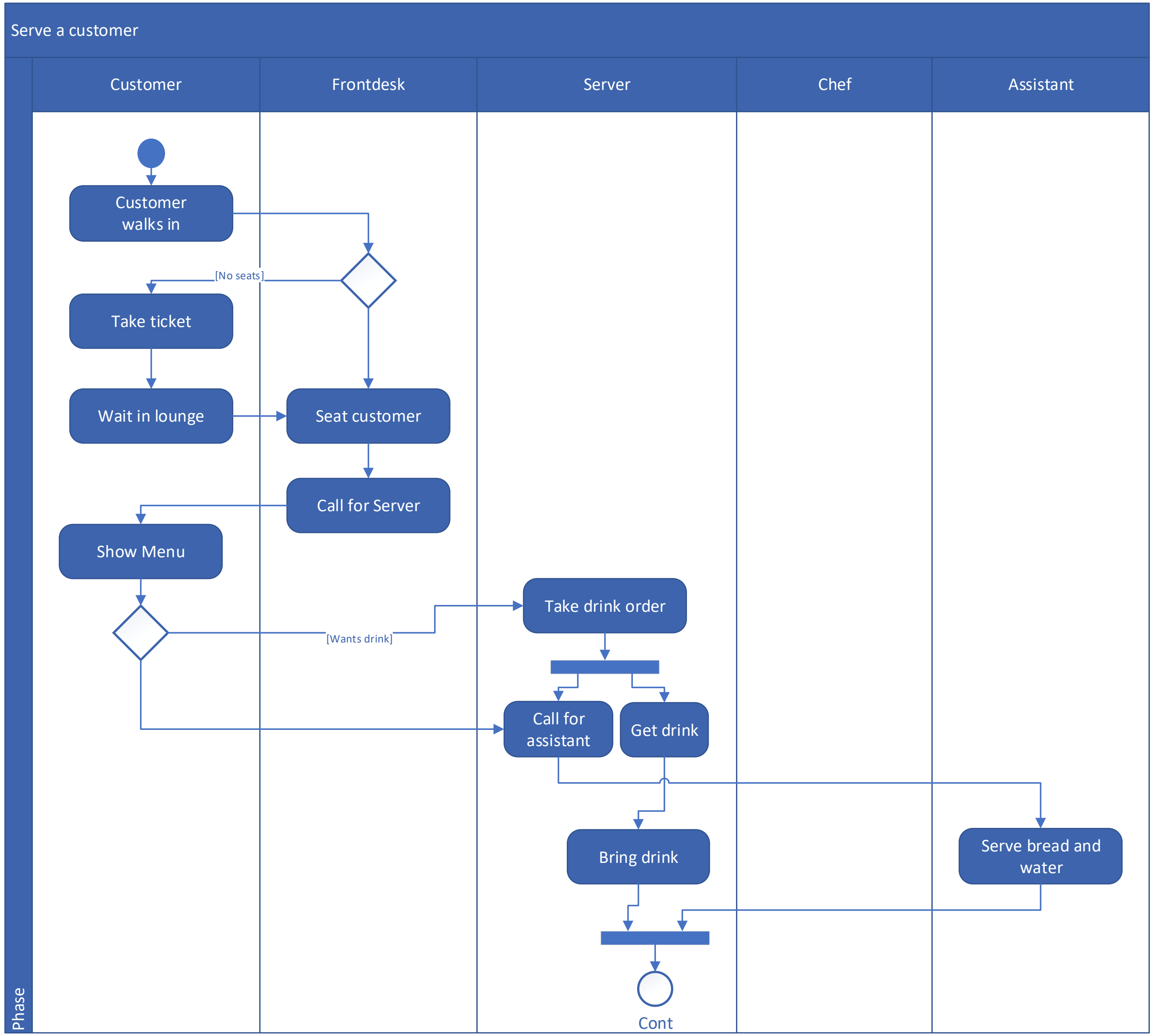
Customer

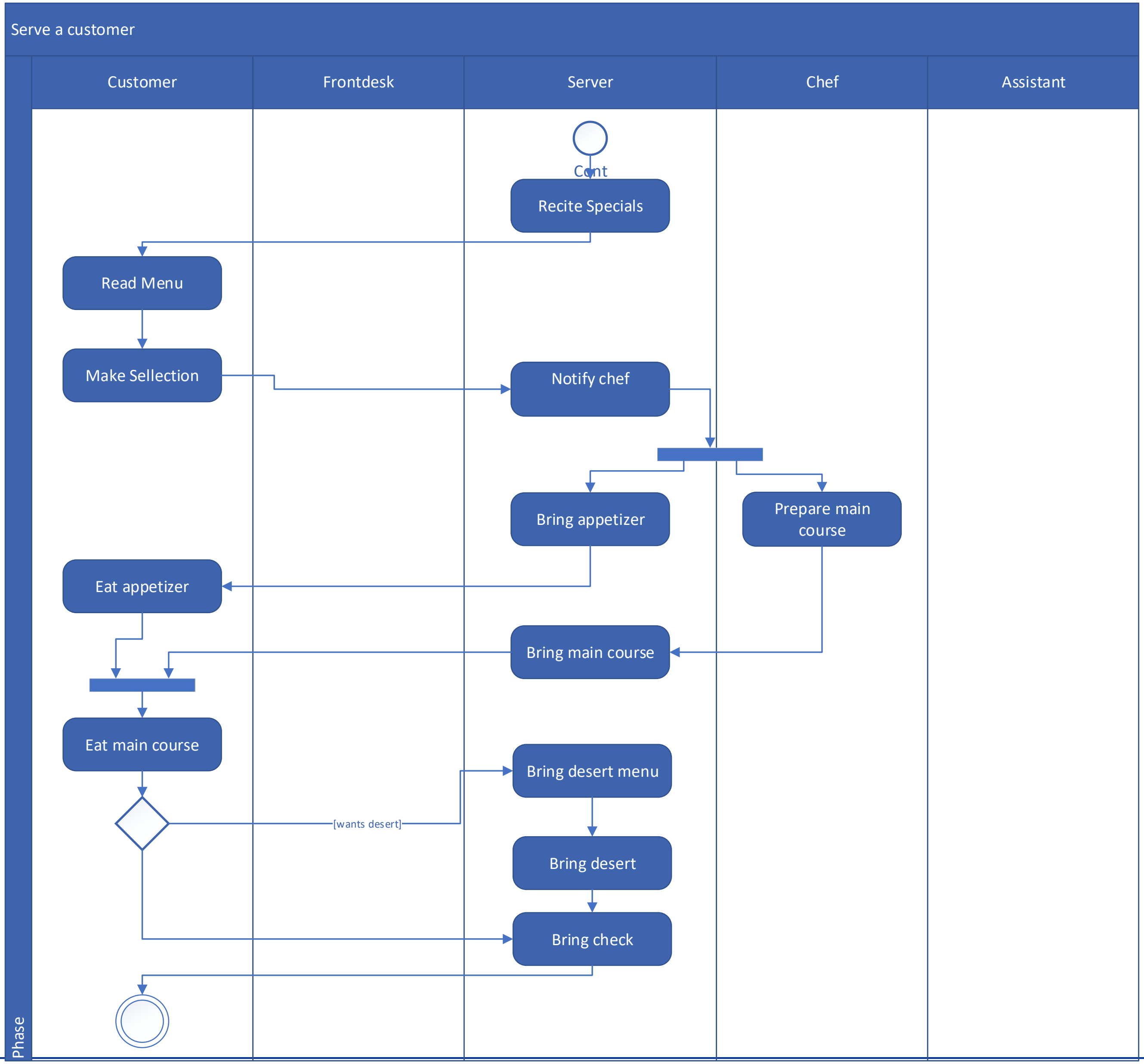
Chef

Server

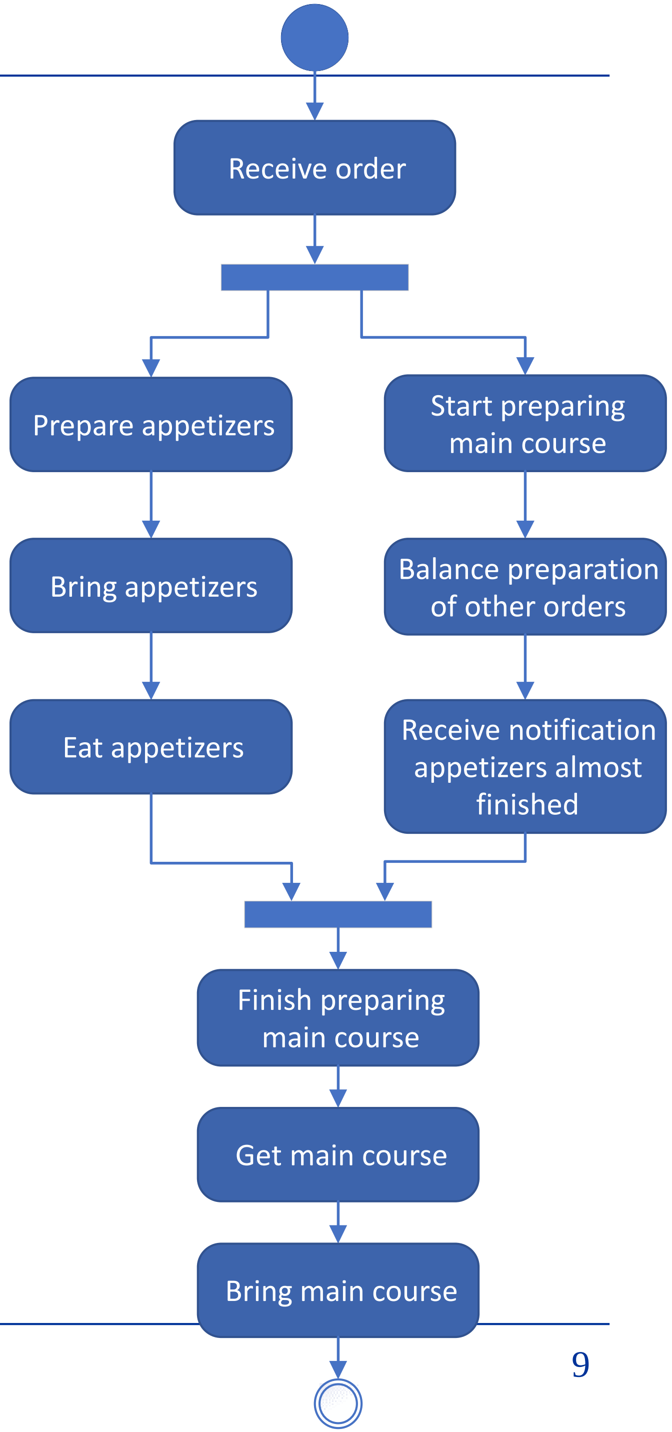
Discover Domain Procedures

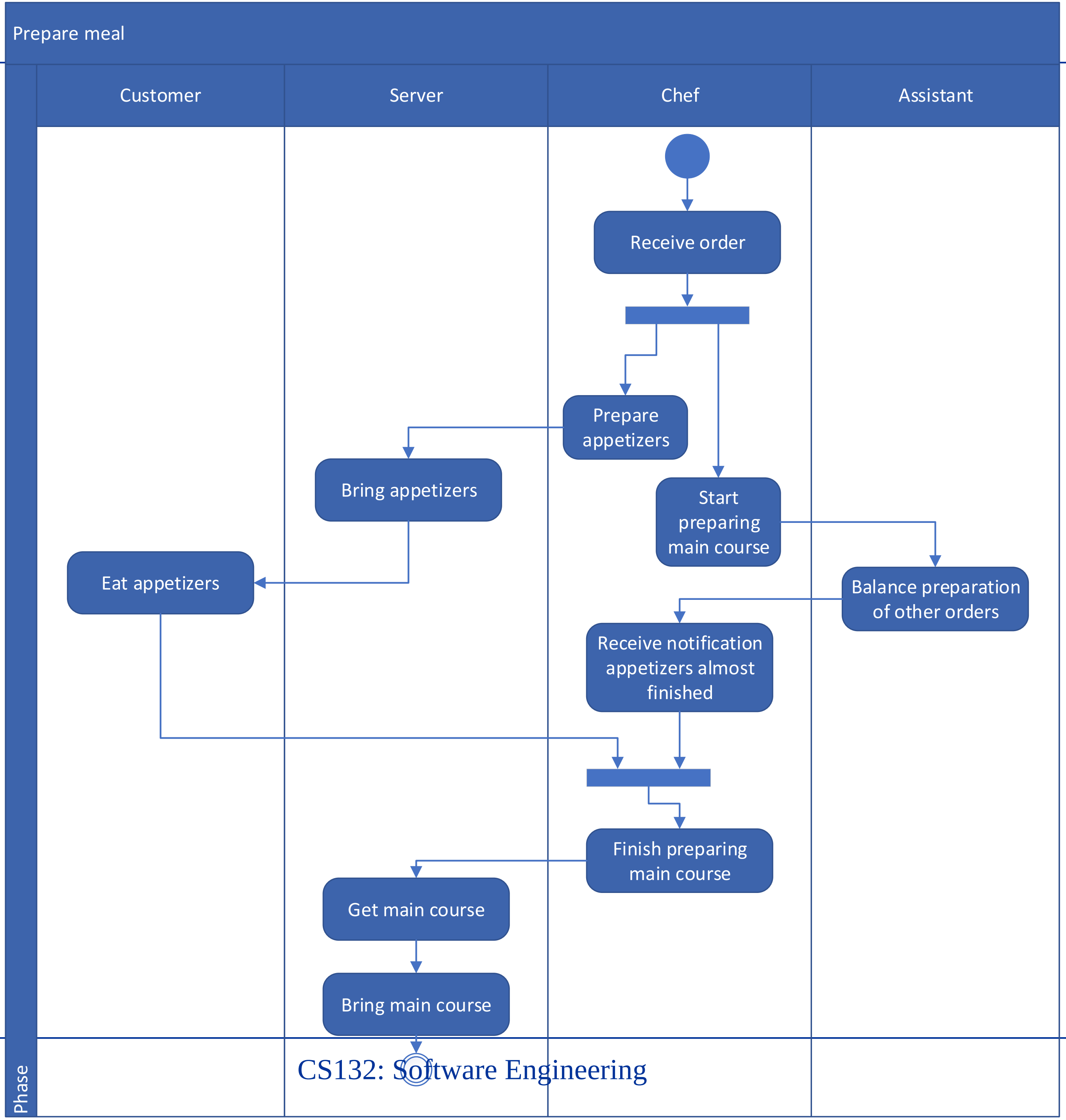



Serve a customer
Phase
Customer
Frontdesk
Server
Chef
Assistant
Customer walks in
[No seats]
Take ticket
Wait in lounge
Seat customer
Call for Server
Show Menu
[Wants drink]
Take drink order
Call for assistant
Get drink
Bring drink
Serve bread and water
Cont

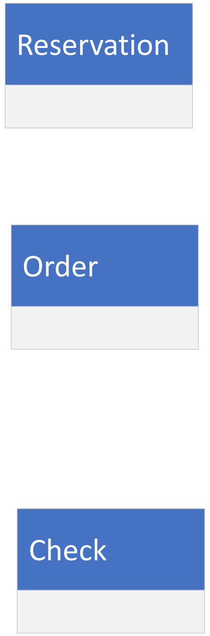

Serve a customer
Customer
Frontdesk
Server
Chef
Assistant
Phase
Cont
Recite Specials
Read Menu
Make Sellection
Notify chef
Bring appetizer
Prepare main course
Eat appetizer
Bring main course
Eat main course
Bring desert menu
[wants desert]
Bring desert
Bring check

- Prepare food



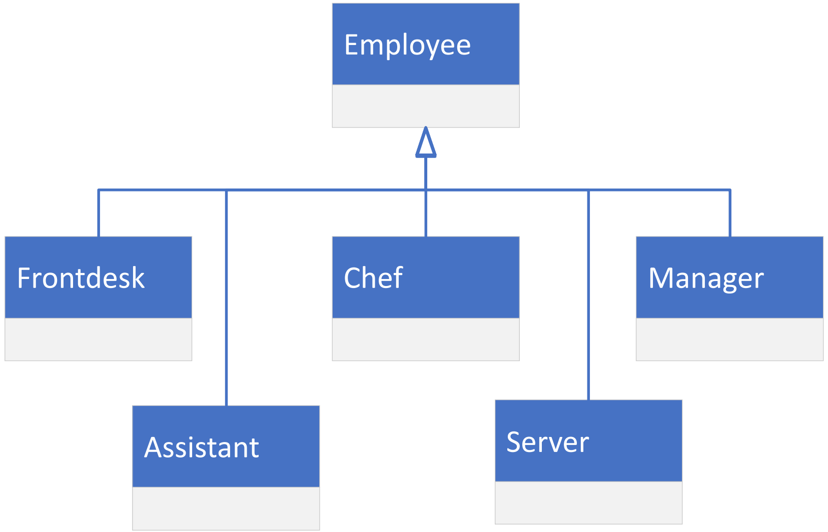

Prepare meal
Customer
Server
Chef
Assistant
Receive order
Prepare appetizers
Bring appetizers
Start preparing main course
Eat appetizers
Balance preparation of other orders
Receive notification appetizers almost finished
Finish preparing main course
Get main course
Bring main course
Phase

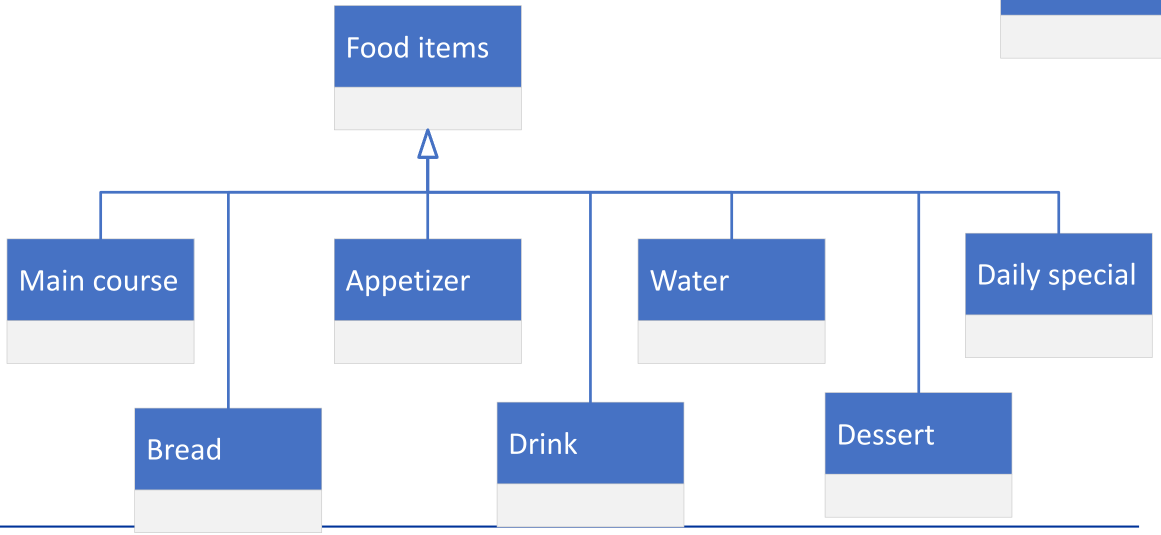
Domain Analysis



1. Develop 1st version of class diagram

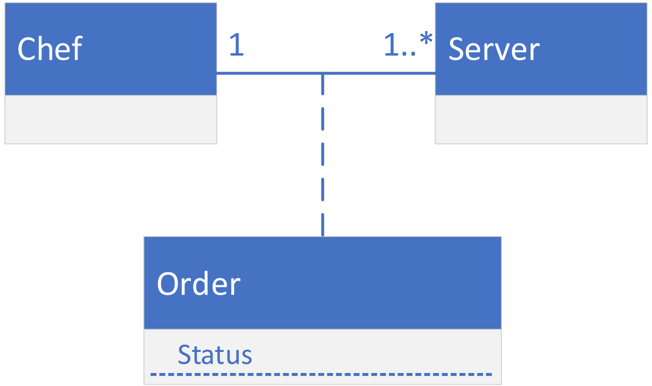
2. Find similar attributes and organize objects into classes

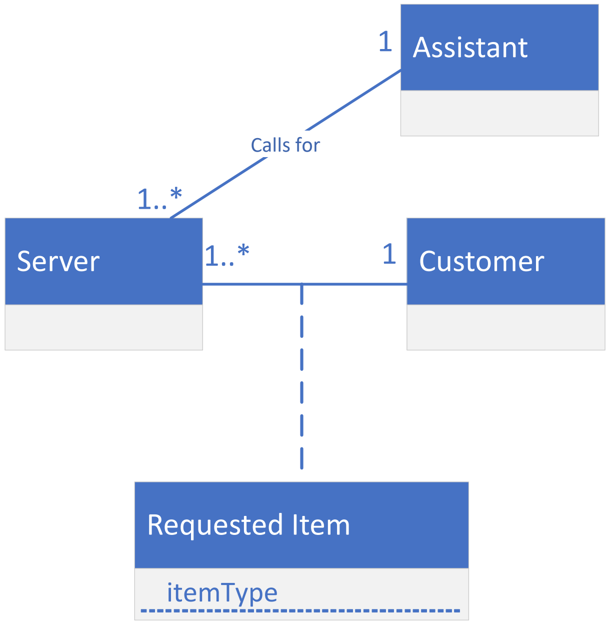


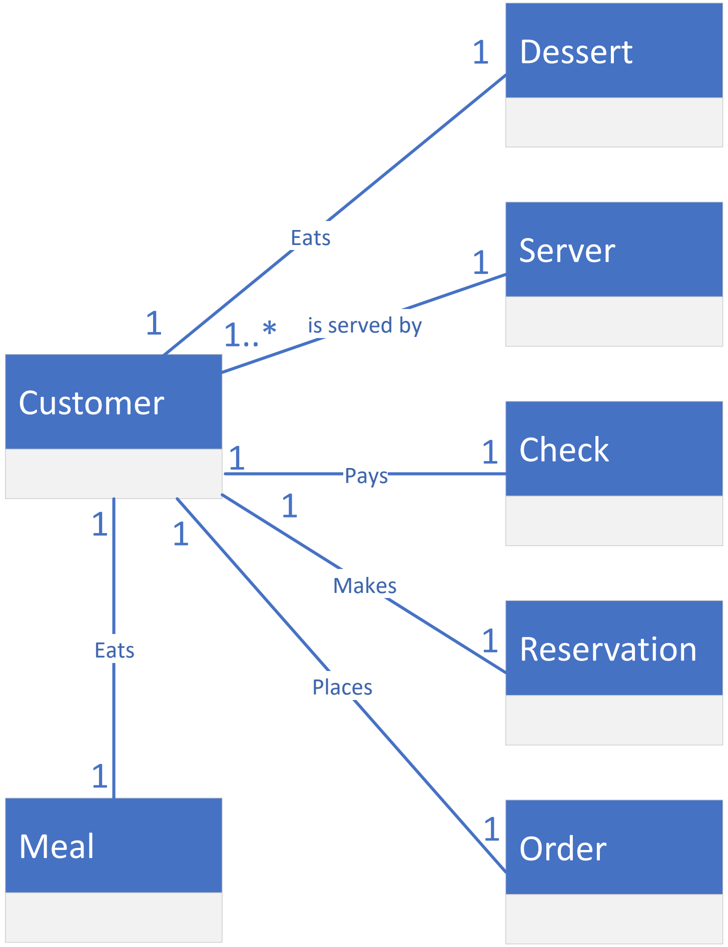


Domain Analysis (cont.)

3. Further understand the domain
   – Find associations

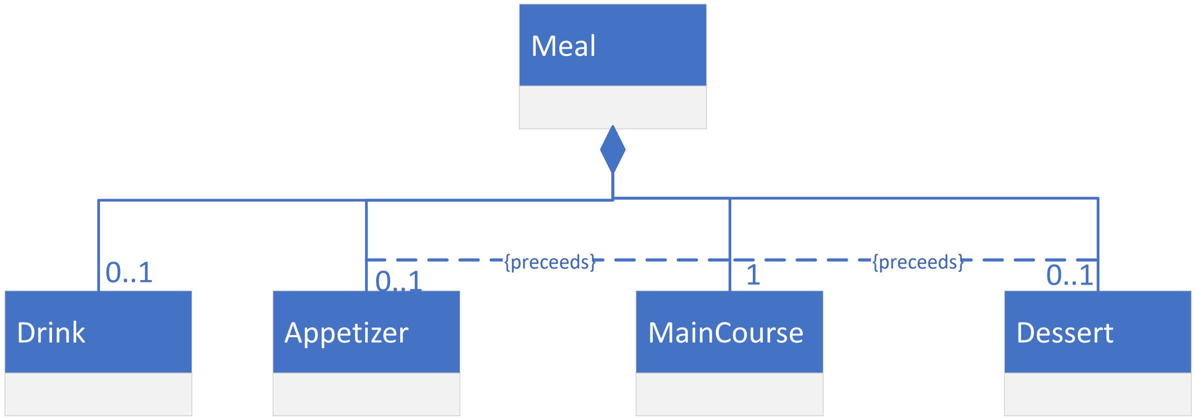


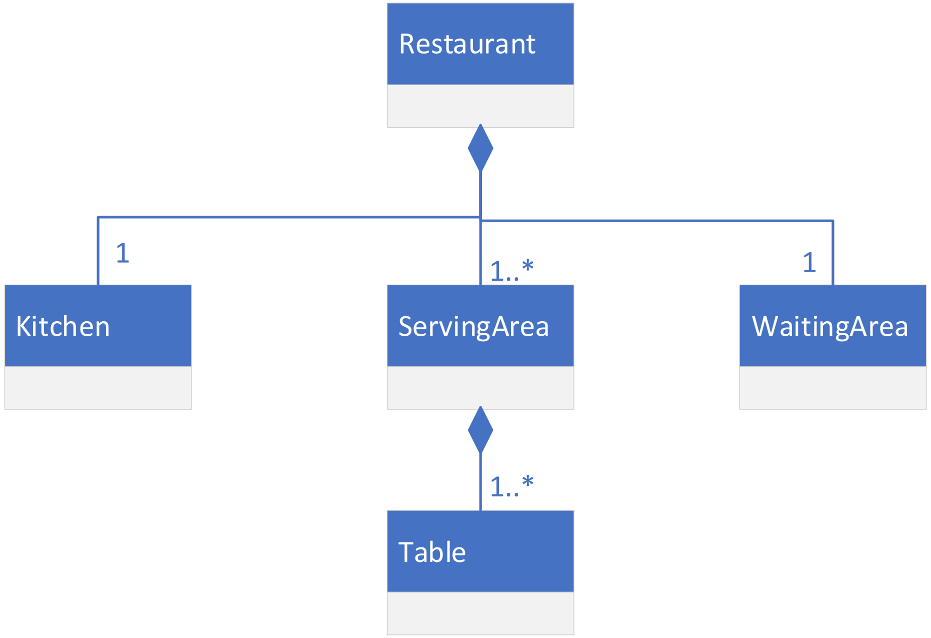




Domain Analysis (cont.)

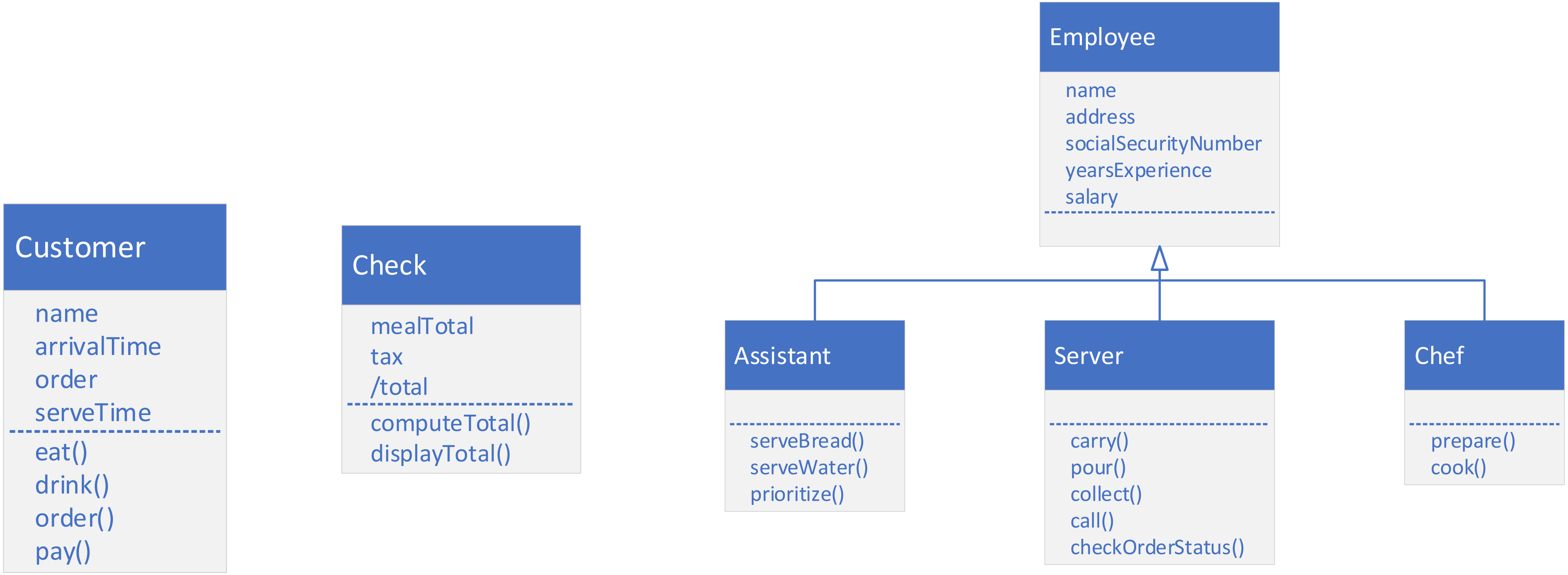
4. Find aggregations and compositions





Domain Analysis (cont.)

5. Enrich information in classes



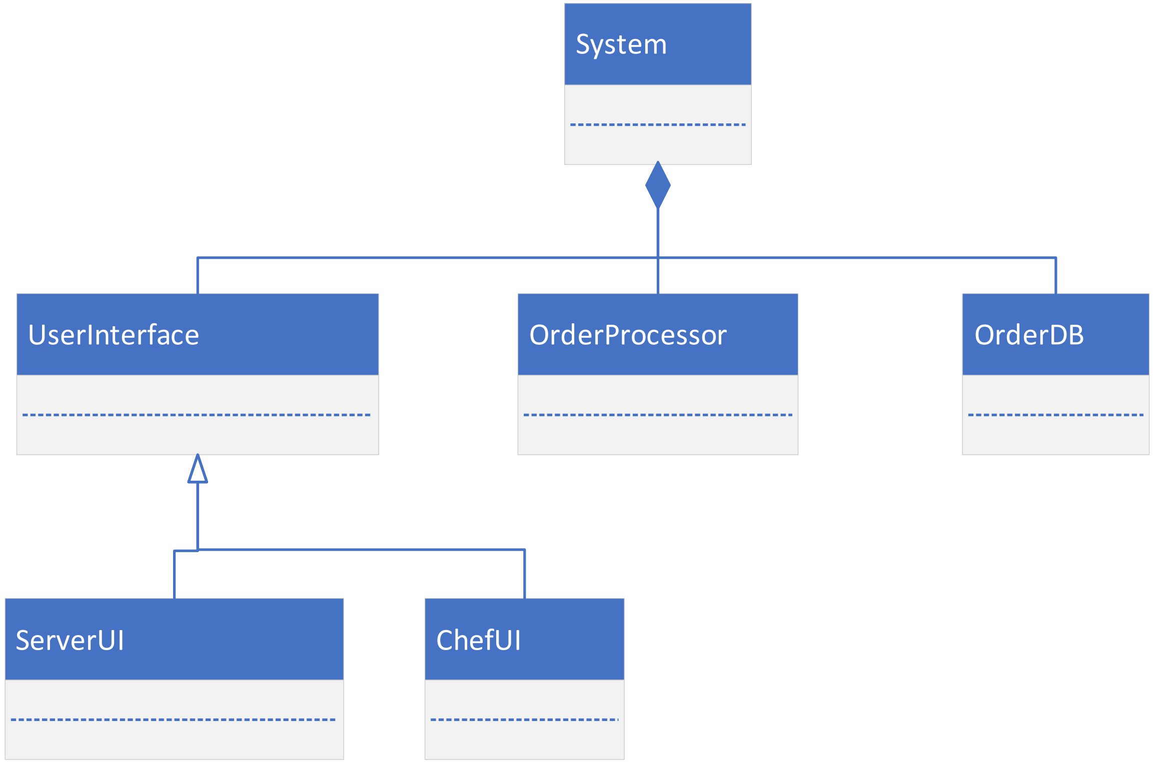
Discover system requirements

- Joint Application Development (JAD) session
  - Restaurant owner
    - Understands the overall objectives of the system
  - Server
    - Actual user of the system
  - System analyst
    - From solution’s perspective: propose potential system architecture
  - Modeler
    - From problem’s perspective: abstract potential use cases
  - Coordinator
    - Keep the conversations on track

Discover system requirements (cont.)

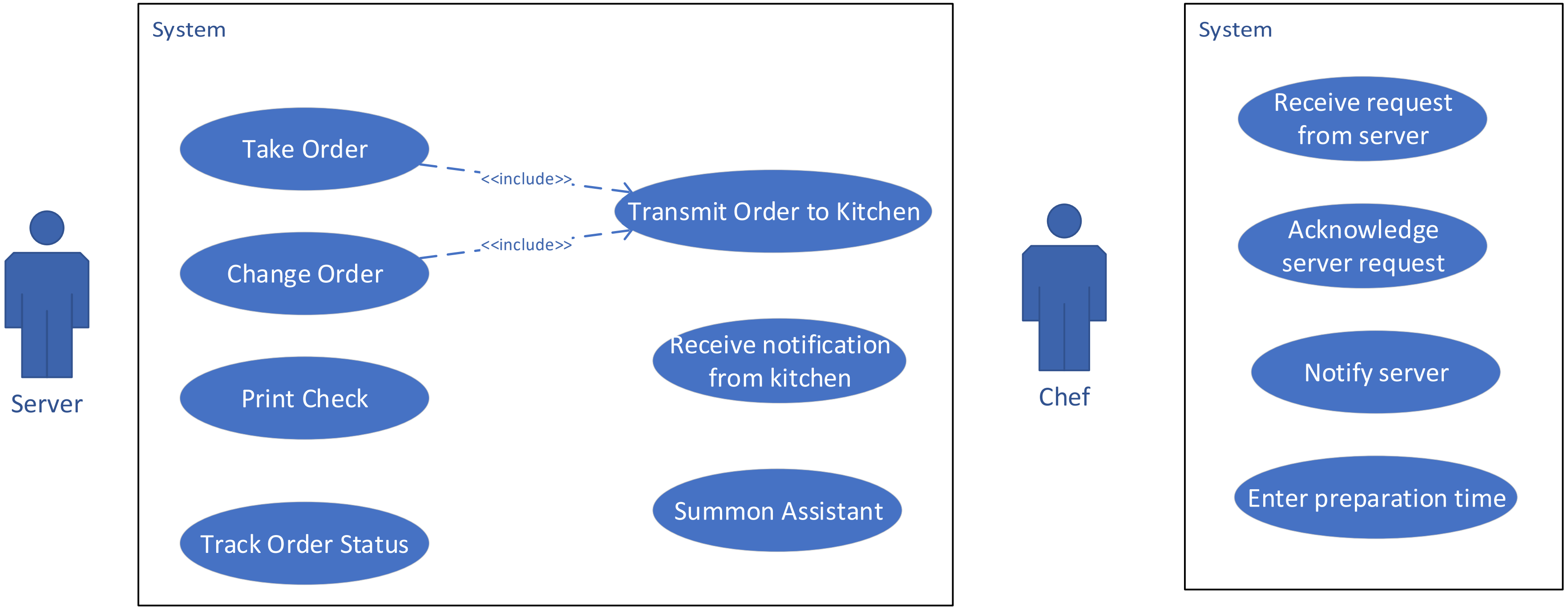
- Requirements for intelligent restaurant system
  - Primary: Save the server's travel time between kitchen and serving area
  - Secondary: Improve serving quality and efficiency

- Proposed solution
  - An order database that keeps track of order information
  - An order processor that handles order generation/modification
  - User interface for both the chef and the server



Discover system requirements (cont.)

- System requirements as use cases

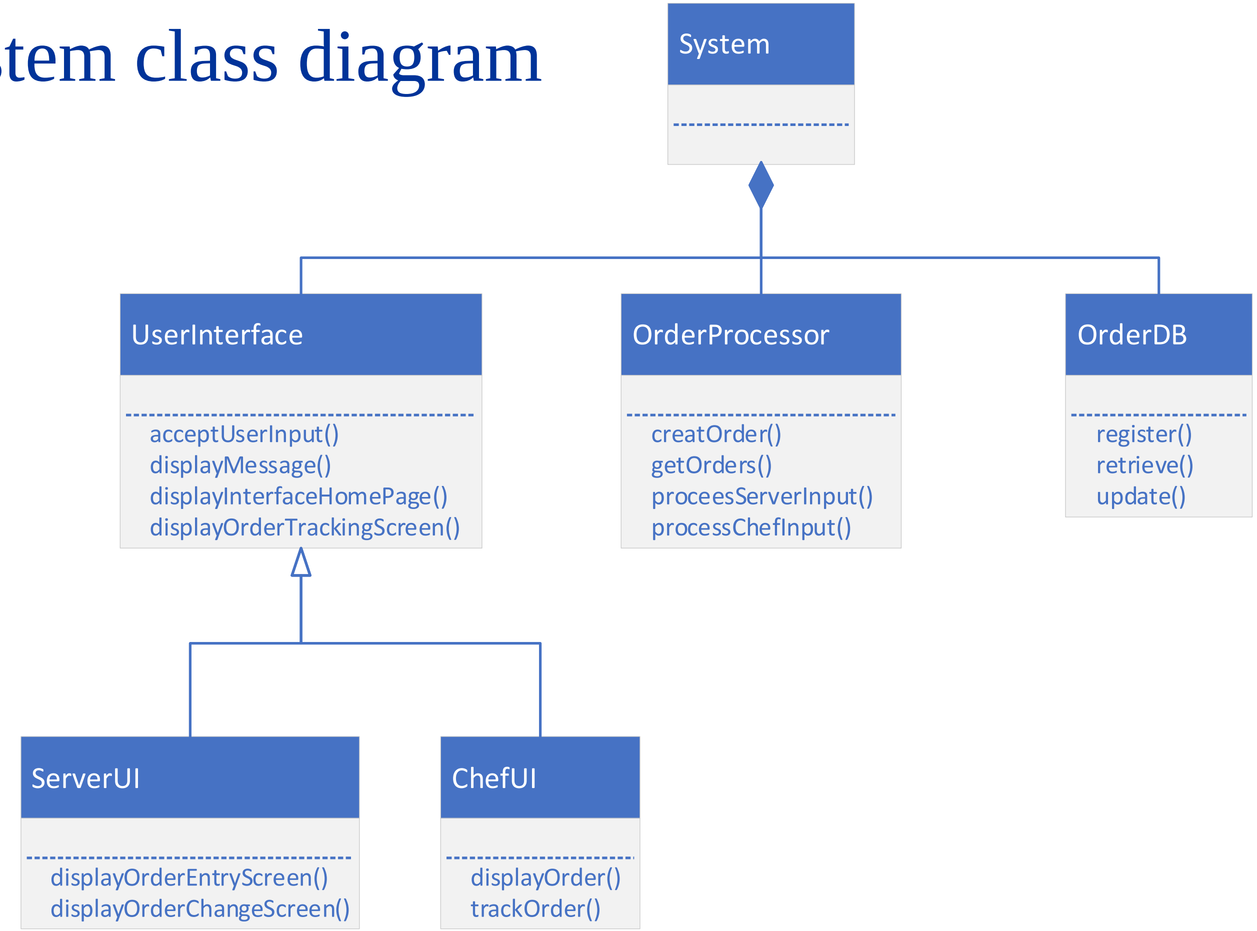


Discover system requirements (cont.)

- Expanding use cases in another JAD meeting
- Use case "Take an order"
  - Description: Server inputs order data in his/her terminal and transmit the order to the kitchen.
  - Precondition: Customer has read the menu and made selections
  - Postcondition: Order has been input into the system
  - Standard procedure
    1. Server activate the order entry screen on his/her terminal
    2. Server input the order information on the screen
    3. System send the order to the chef UI

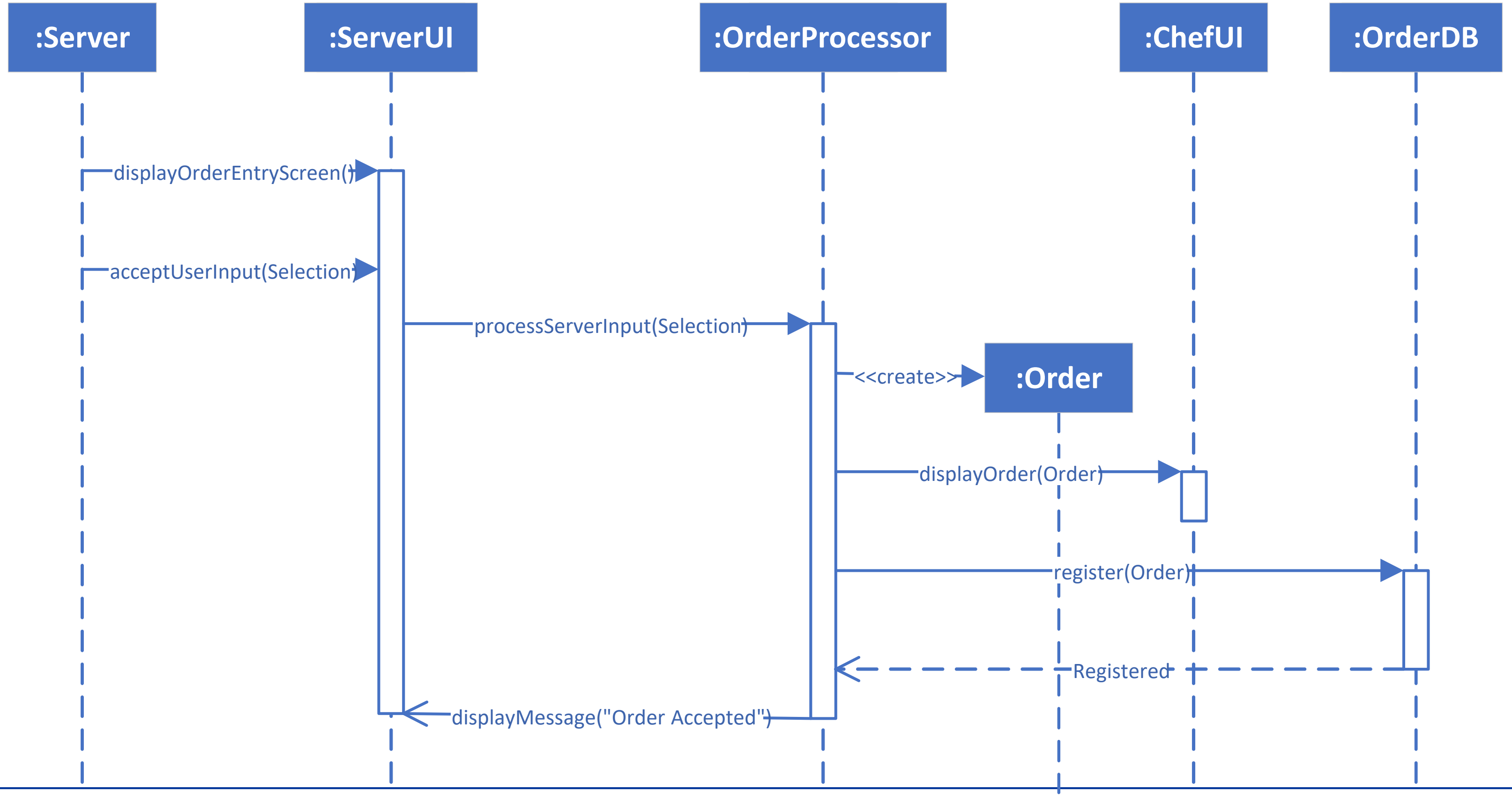
Discover system requirements (cont.)

- Enriched system class diagram



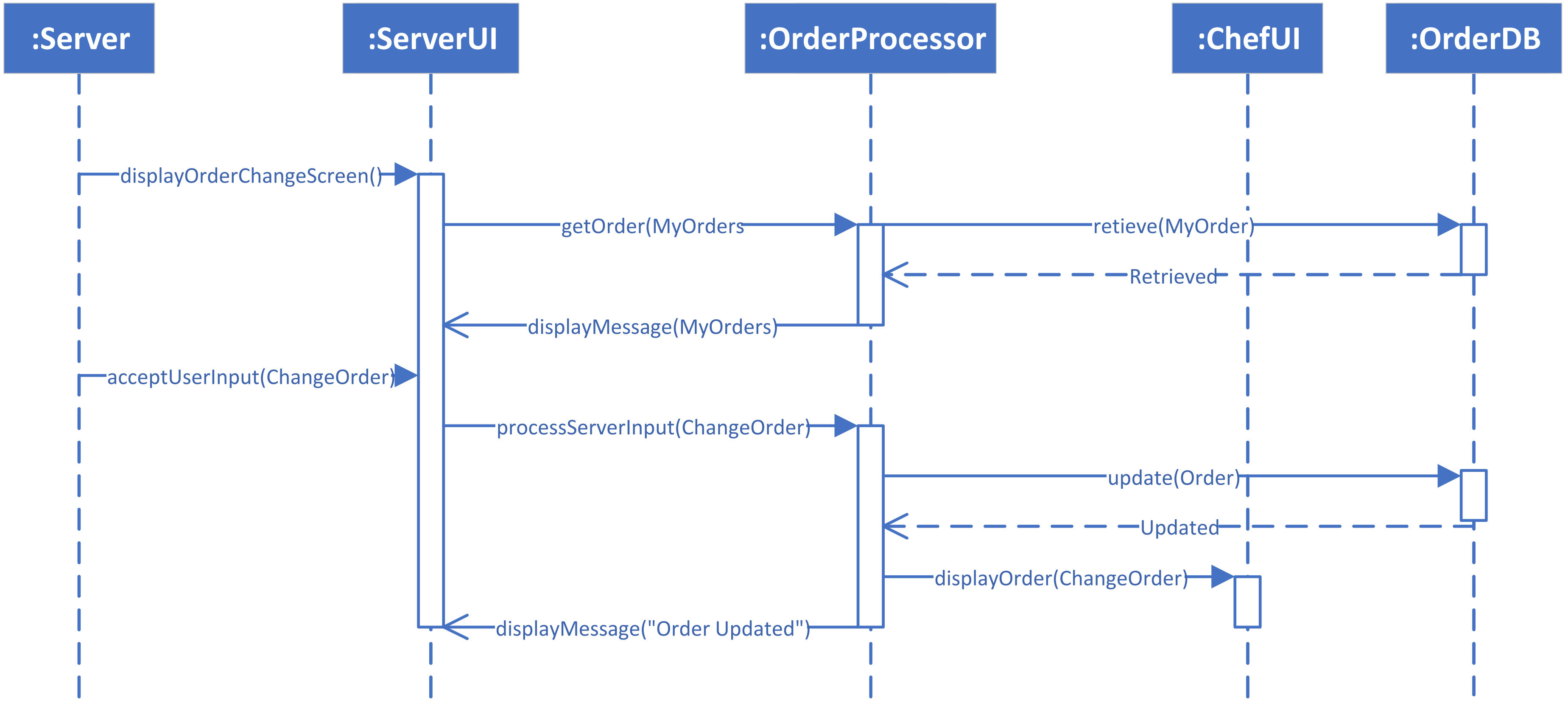
Identify interactions

- Use case “Take an order”



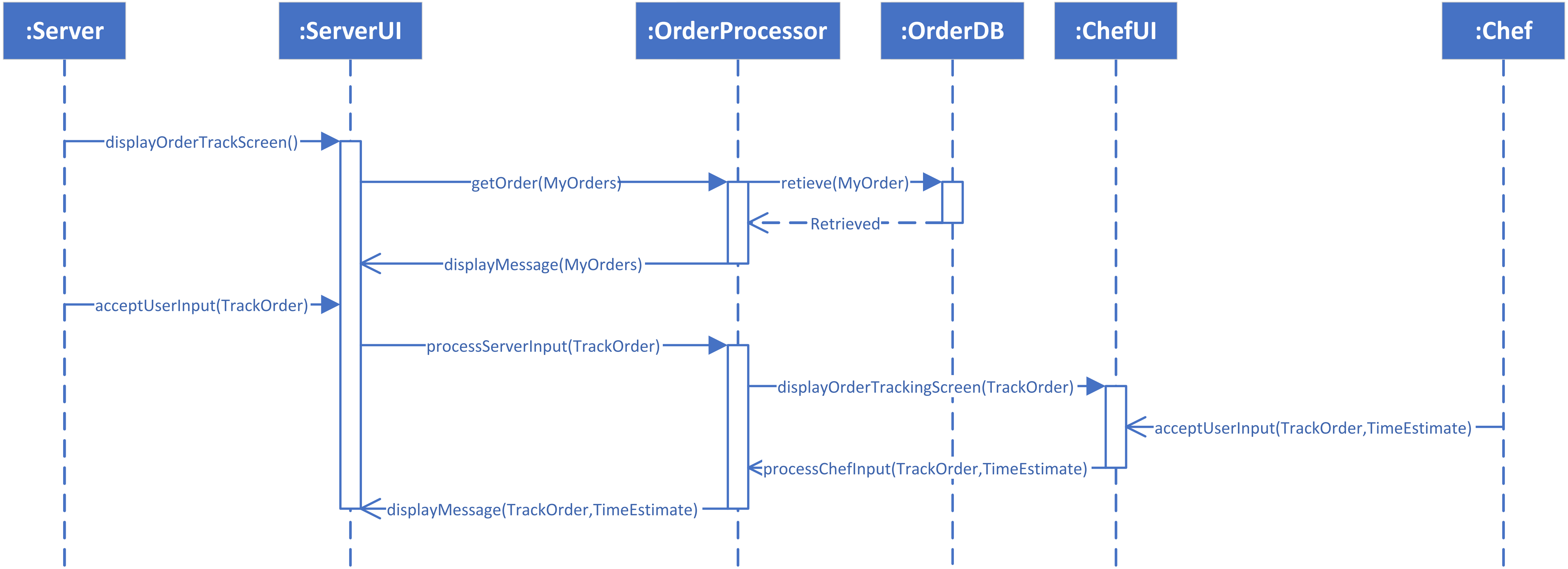
Identify interactions (cont.)

- Use case “Change an order”



Identify interactions (cont.)

- Use case “Track an order”



Why do we need models?

- Prediction
  - We know the low-level mechanisms but we want to understand how they affect higher-level behaviors
  - Use simulation instead of testing on the real system

- Explain the data
  - Make assumptions and use our knowledge to explain mechanisms that we don’t understand

- Classification
  - i.e. definitions, machine learning algorithms

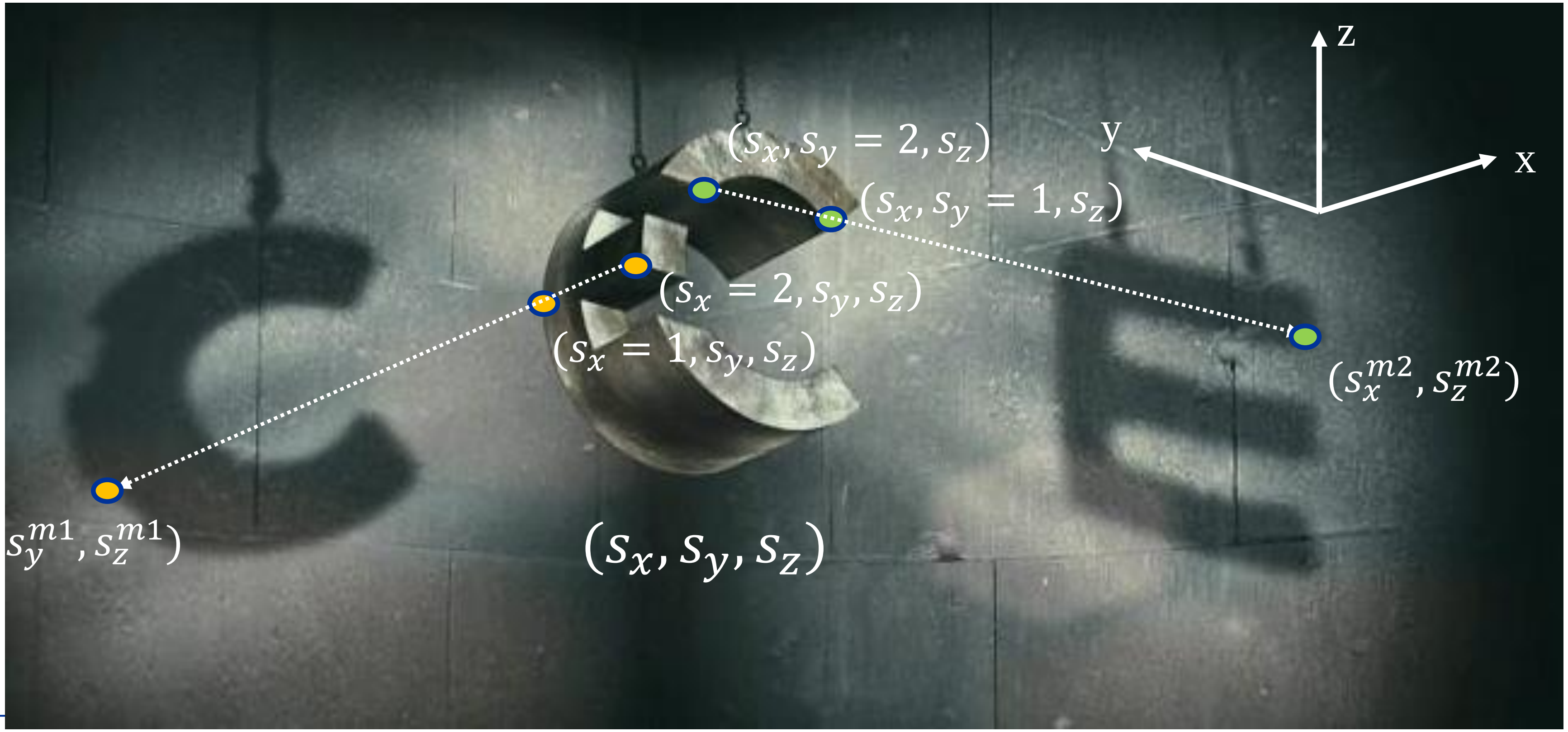
What are models?

- A system: (S, I, T, O)
  - S: States $s_1, s_2 \dots s_n$
  - I: Inputs (could be $\emptyset$)
  - T: Transitions $S \times I \times S$
  - O: Observations $f(S_o), S_o \subseteq S$

- Model of the system (S^m, I^m, T^m)
  - S^m: Abstraction/interpolation of S
    - Much fewer state variables
  - I^{m}: abstraction of I (could be $\emptyset$)
  - T^m: Transitions $S^m \times I^m \times S^m$

Abstraction – removal of state variables

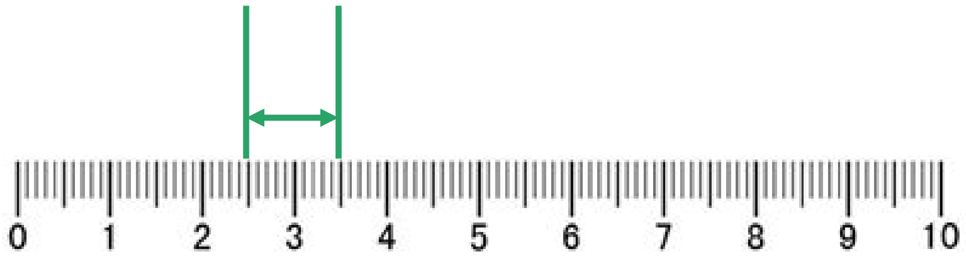
- States (s_x, s_y, s_z) *are abstracted to* (s_y^{m1}, s_z^{m1})
  - $(s_x, s_y, s_z) \rightarrow (s_y^{m1}, s_z^{m1})$
- Loss of information



Abstraction: Approximation of state variable values

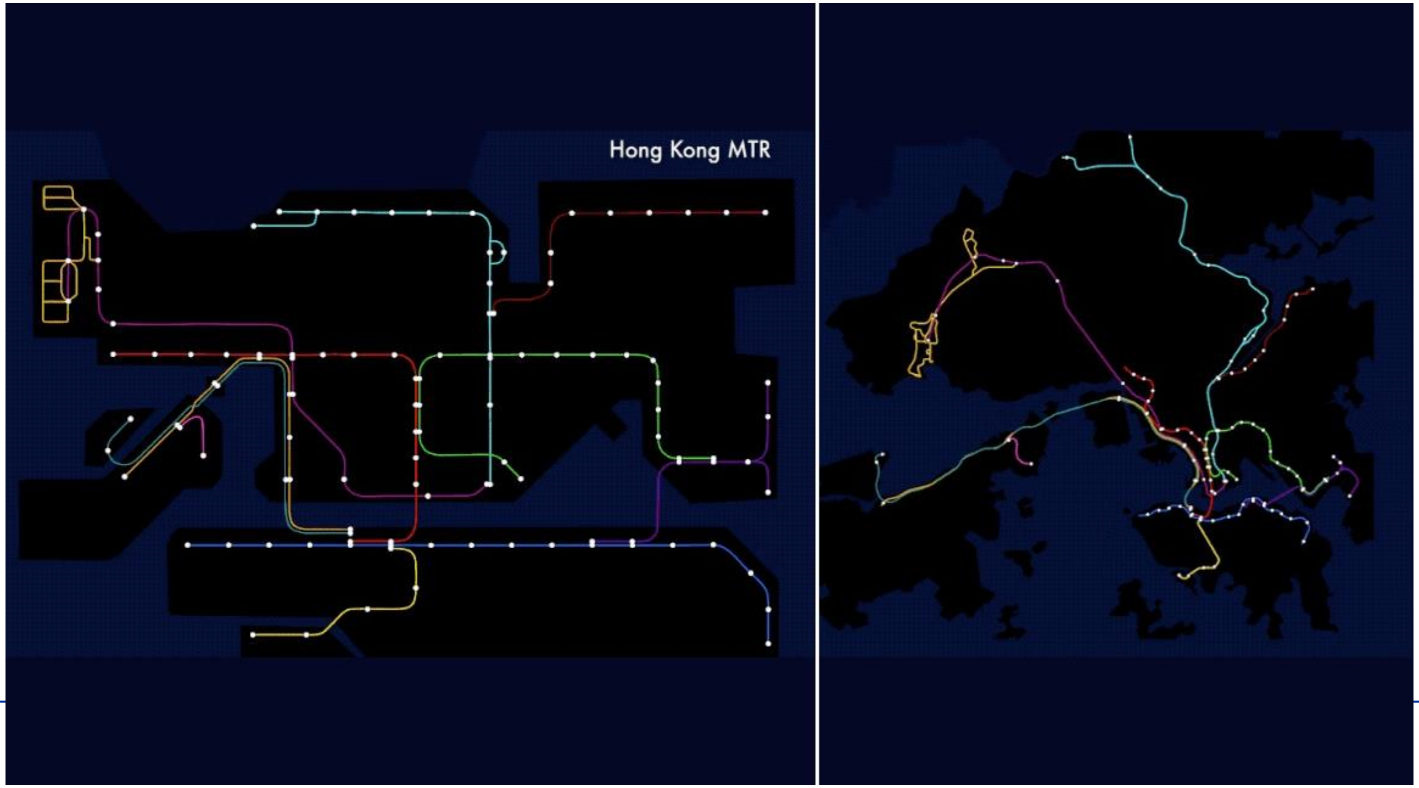
- Irrational numbers
  - $\pi \approx 3.1415$
  - $\sqrt{2} \approx 1.414$

- Approximation is another way of abstraction

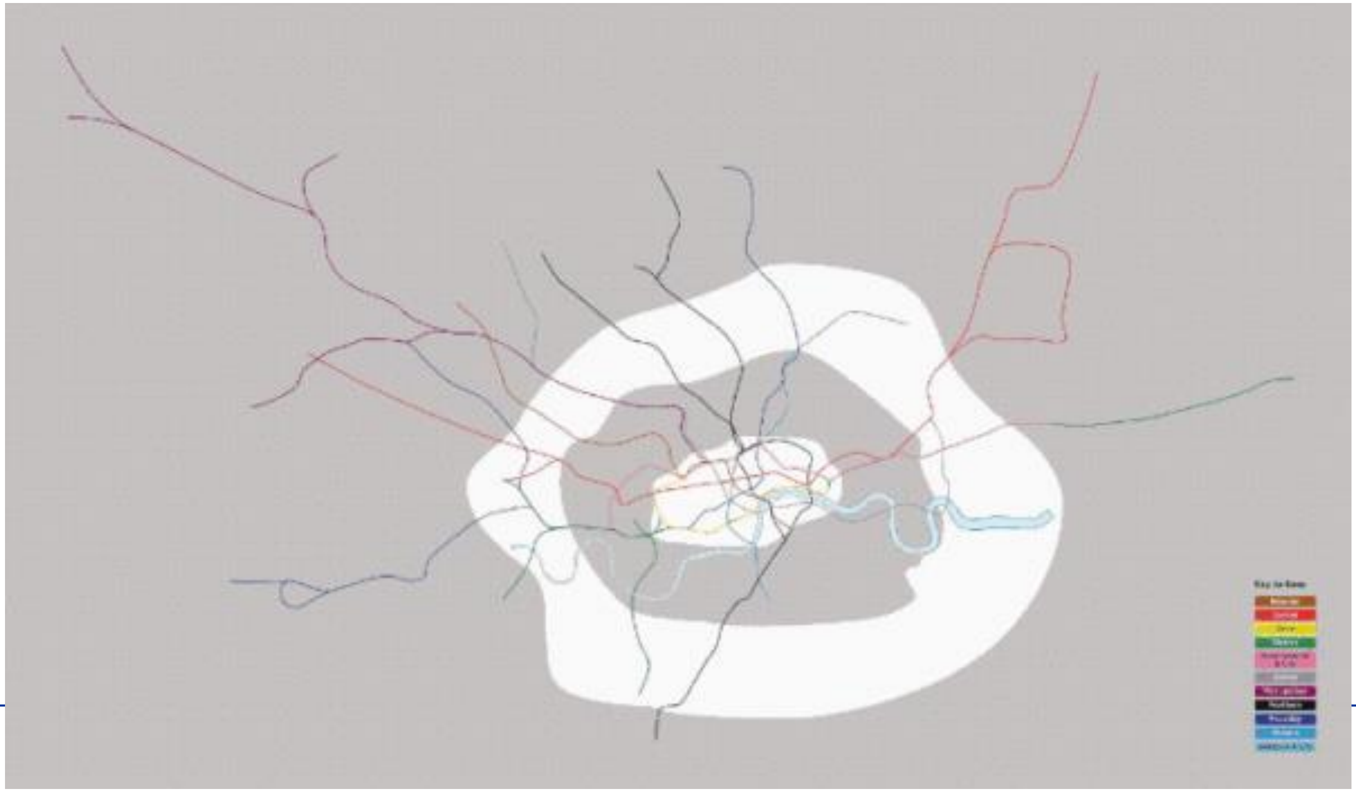


Interpolation: extracting interpretable information

- Locational information -> topological information
- $S^m = f(S_p), S_p \subseteq S$



More Interpolation: London MTR



What is considered as a "good" model?

- Accuracy
  - All models are wrong!
  - Error accumulates over time
  - Initial condition of the model cannot be determined due to limited observability
- Generality
  - The capability to explain not only training data, but also testing data
- Identifiability
  - Model parameters can be identified from data
- Interpretability
  - S^m are meaningful and interpretable by human

Newton vs. Einstein

- Newtonian physics is suitable for macro level objects at low speed
- $L = L_0\sqrt{1-\frac{v^2}{c^2}}$
- A model can only be "good" within the context of its designated application
- The definition of "goodness" is changing over time

Modeling methodologies

- Bottom-up modeling
  - "White-box" model
  - Using first principles
  - Pros:
    - Interpretable
    - Convincing
  - Cons:
    - State space explosion
    - Difficult to be general
    - Low identifiability

- Data driven models (i.e. Neural networks)
  - "Black-box" model
  - From observable data
  - Pros:
    - No need to know domain knowledge
  - Cons:
    - Large and uninterpretable S^m
    - Depends highly on the quality and quantity of data