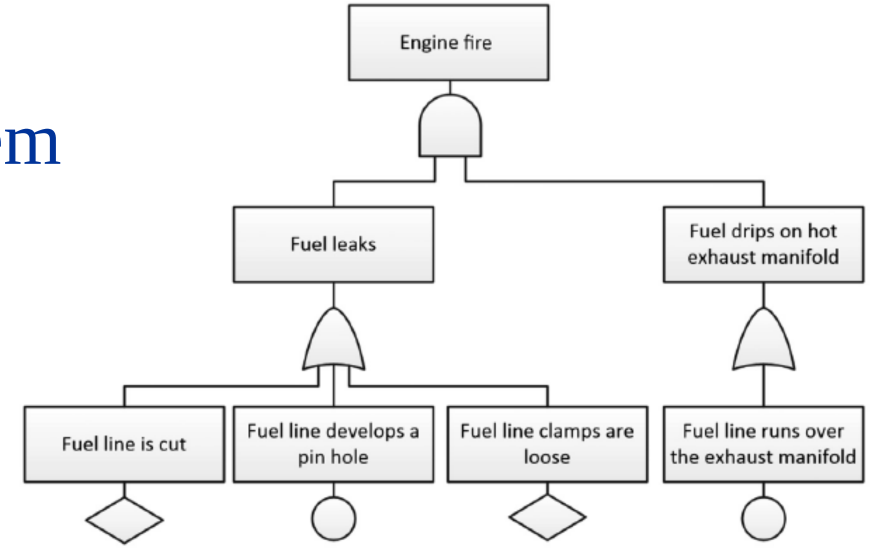
Lecture 7: Applications of UML

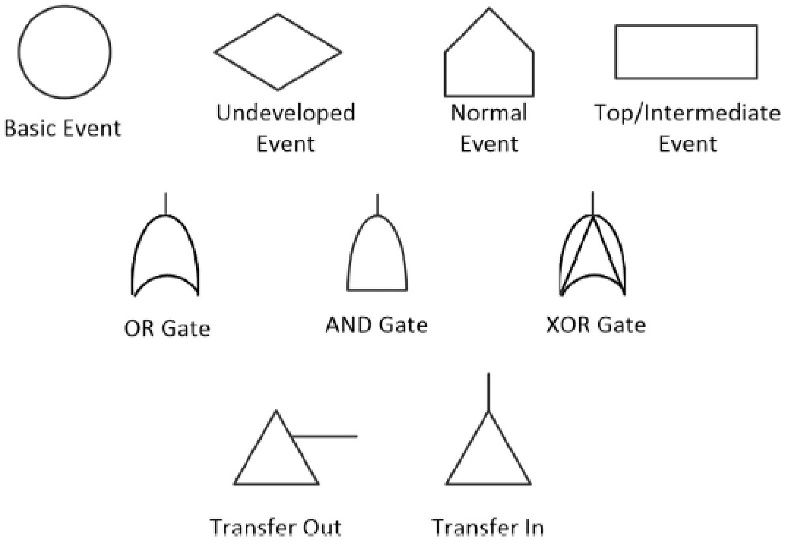
Fault Tree Analysis (FTA)

- Developed by Bell Labs in 1962 during development of the Minuteman missile system
- Later been used in Nuclear Regulation and also NASA
- A deductive top-down reasoning process
  - Starts from the undesired system outcomes
  - Attempts to find out all the credible sequences of events that could result in the undesired system outcomes



Fault Tree Analysis Symbols

- Basic Event:
  - Requiring no further development
- Undeveloped Event
  - An event that is not further developed due to lack of information, or when the consequences are not important
- Normal Event
  - An event that is normally expected to occur, e.g., the device gets used
- Top/Intermediate Event
  - An event that is further analyzed
- OR Gate
  - Output occurs when one or more of the inputs occur
- AND Gate
  - Output occurs when all of the inputs occur
- XOR Gate
  - Output occurs when only one of the inputs occurs
- Transfer
  - Used to manage the size of the tree on a page, and to avoid duplications

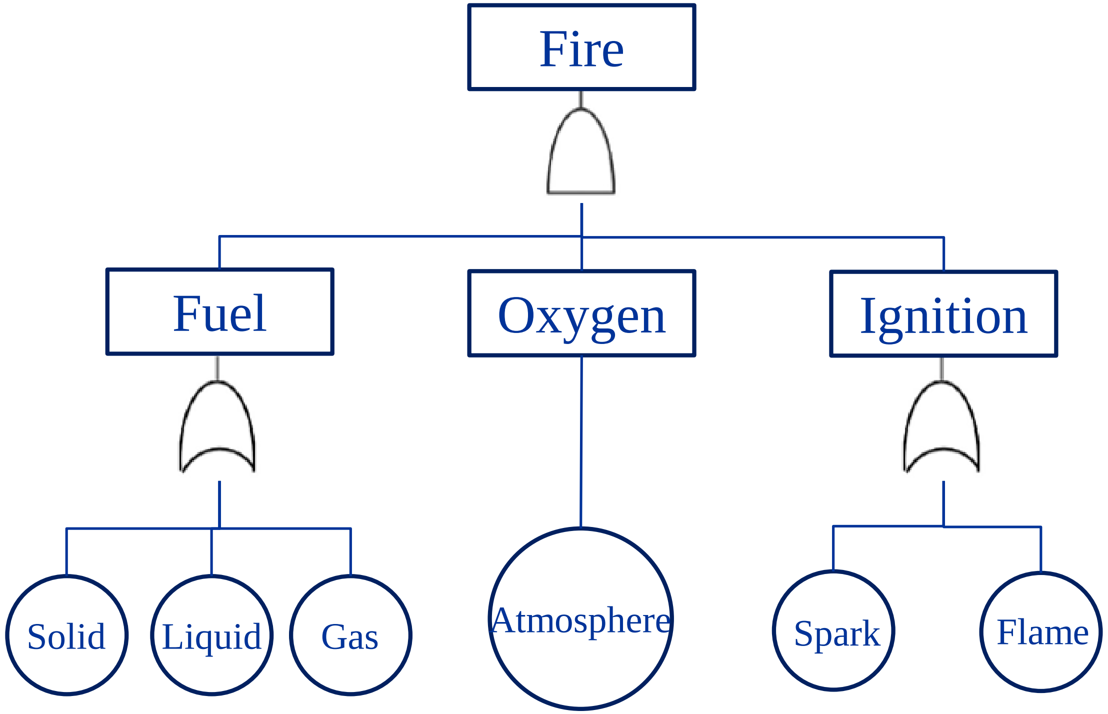


Principles When Identifying the Next Level

- Immediate
  - Is the next event on the lower level, immediately preceding the event in question?
  - Think small/myopically
- Necessary
  - Is the next event on the lower level necessary to cause the fault in question?
  - Avoid entering unnecessary events
- Sufficient
  - Do you have all the necessary events to cause the fault in question?
  - Ensures the higher event can actually happen, given the lower-level events.

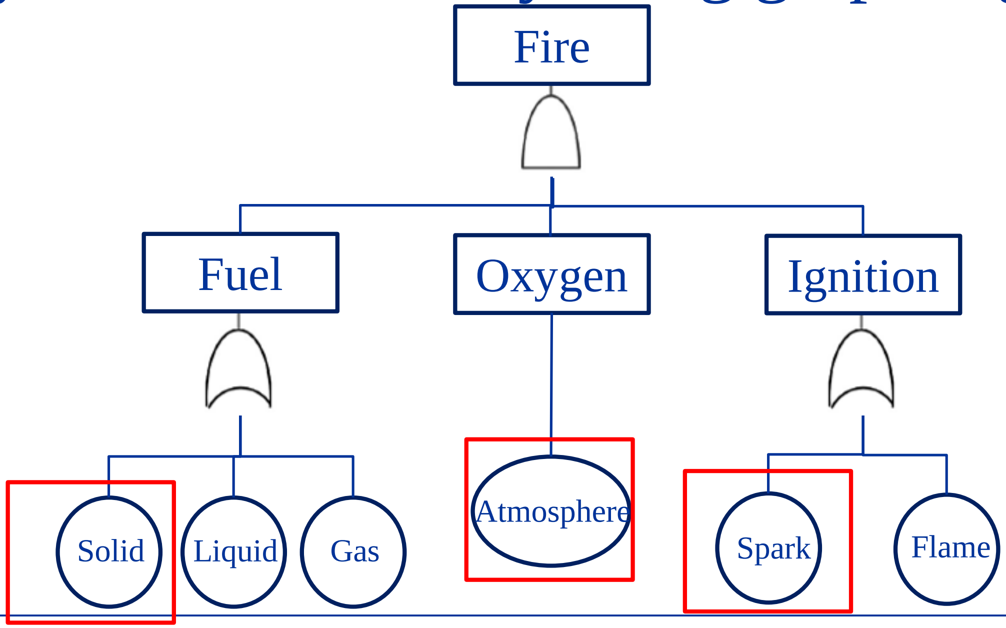
Example

- The hazardous situation "Fire"
  - Requires "Fuel", "Oxygen" and "Ignition"
  - "Fuel" can be either "Solid", "Liquid" or "Gas"
  - "Ignition" can be done with either "Spark" or "Other flames"



Qualitative Analysis

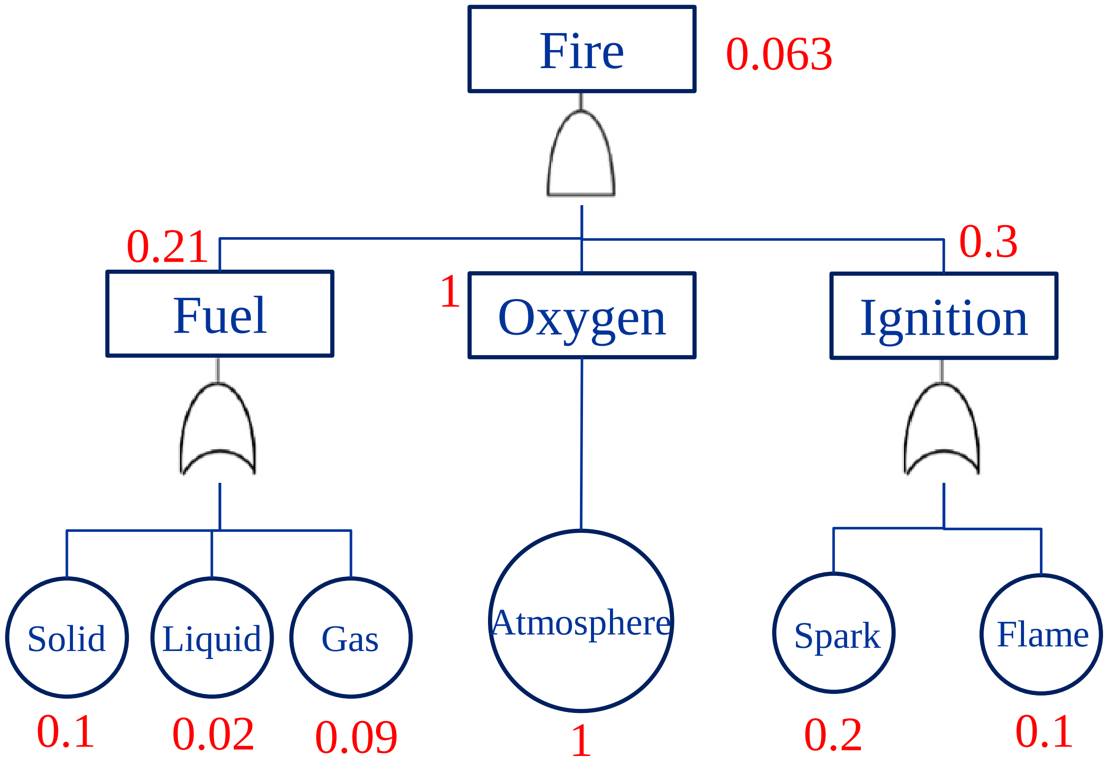
- Minimum Cut Set:
  - The smallest set of basic events, which if they all occur will result in the occurrence of the top event (Not unique)
- Can be analyzed automatically using graph algorithms



Quantitative Analysis

- Probabilities of basic events are measured per certain period
- Add probabilities under the “or” gate
- Multiply probabilities under the “and” gate

- Focus on the “and” gate
  - Only need to bring one down
- Focus on the branches with higher probabilities



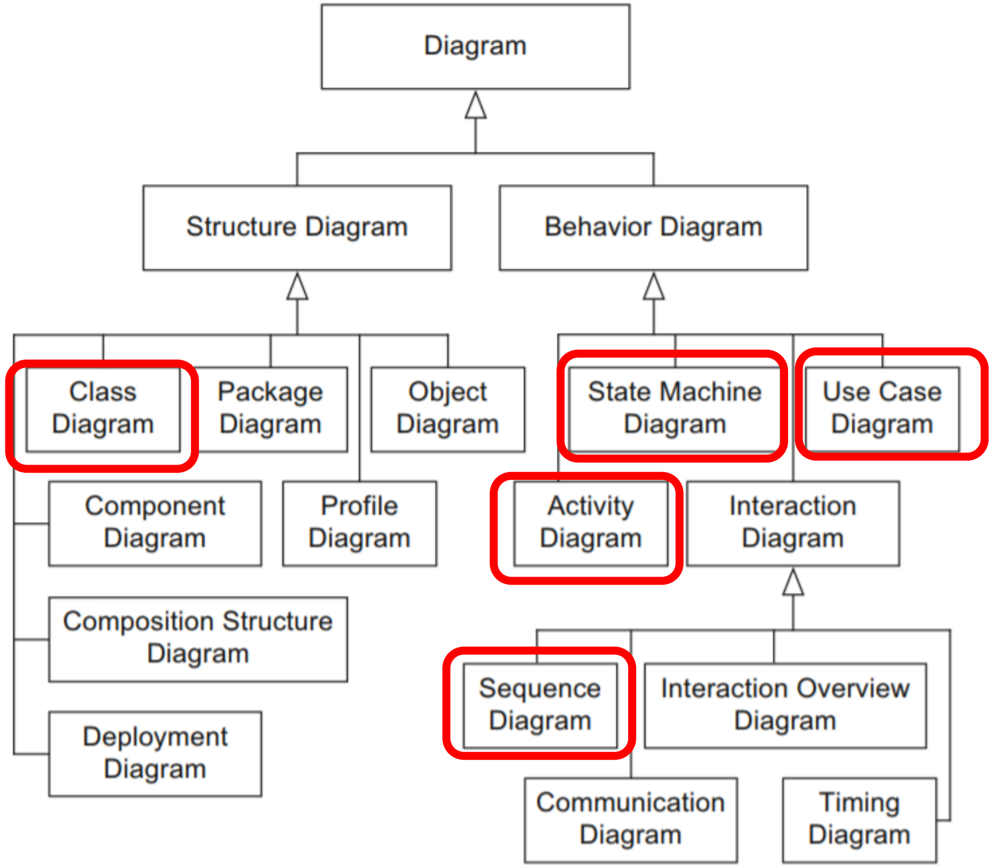
Limitations for FTA

- Can be only used to reason hazardous situations
  - What about other non-safety related qualities i.e. reliability?
- The analysis may not be exhaustive
  - There can be sequences of events that were not identified

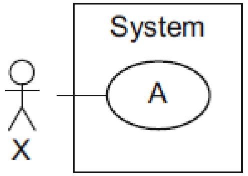
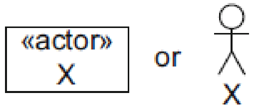
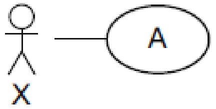
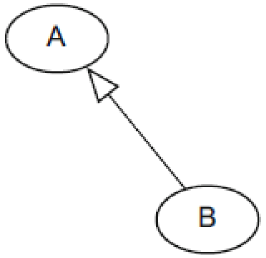
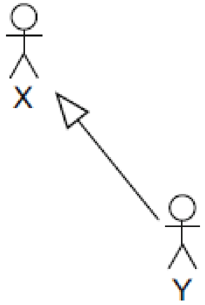
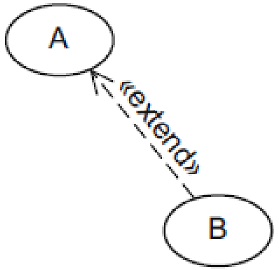
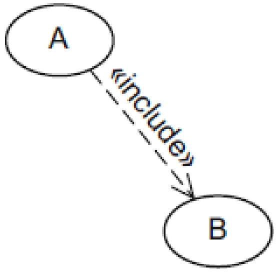
HW 1 is out

- Deadline:
  - 23:59 of specified date.
- Late Policy:
  - Submitted within 24hr after the deadline: 20% discount.
    - For example, if you gained 4 out of 5 points in a homework, with late submission you will only gain 3.2 out of 5 points.
  - After 24hr: 0 points
- Late pass:
  - Each student is given a "Late pass" which can be used ONCE on a homework submission.
  - Extends ONE homework deadline for 24hr, but remember you will still receive penalties if you miss the Extended deadline.

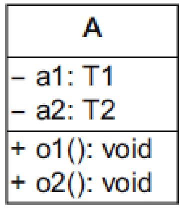
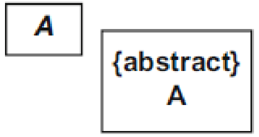
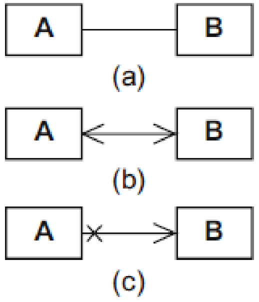
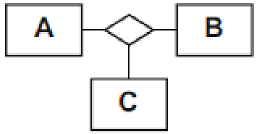
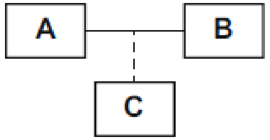
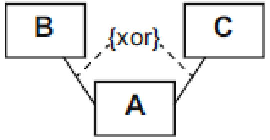
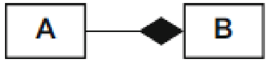
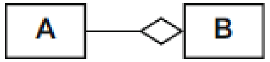
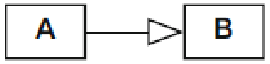
UML Diagrams

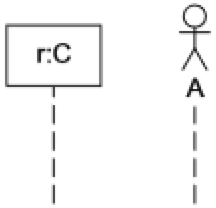
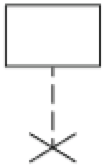
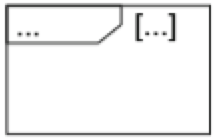
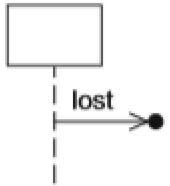
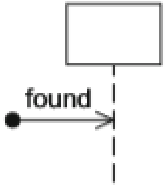


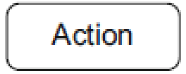
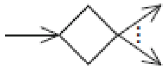
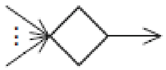
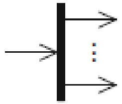
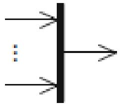
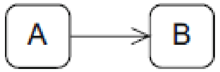
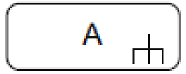
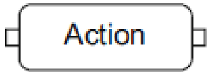
Summary: Use Case Diagram

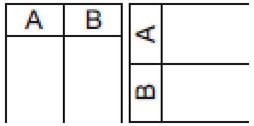
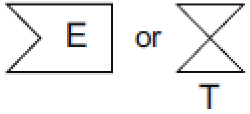
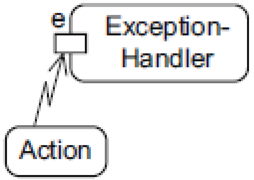
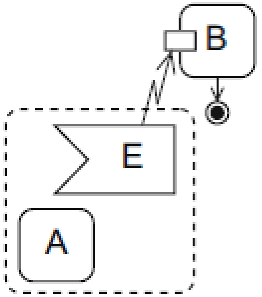
Name	*Notation*	*Description*
System		Boundaries between the system and the users of the system
Use case		Unit of functionality of the system
Actor		Role of the users of the system
Association		X participates in the execution of A
Generalization (use case)		B inherits all properties and the entire behavior of A
Generalization (actor)		Y inherits from X; Y participates in all use cases in which X participates
Extend relationship		B extends A: optional incorporation of use case B into use case A
Include relationship		A includes B: required incorporation of use case B into use case A

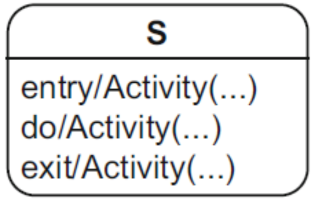
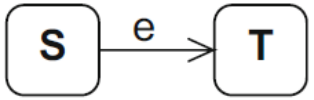
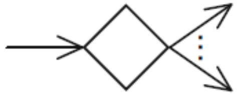
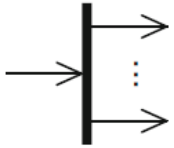
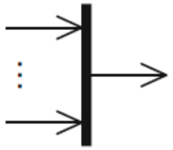
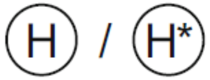
Summary: Class Diagram

Name	*Notation*	*Description*
Class		Description of the structure and behavior of a set of objects
Abstract class		Class that cannot be instantiated
Association		Relationship between classes: navigability unspecified (a), navigable in both directions (b), not navigable in one direction (c)
N-ary association		Relationship between N (in this case 3) classes
Association class		More detailed description of an association
xor relationship		An object of A is in a relationship with an object of B or with an object of C but not with both
Strong aggregation = composition		Existence-dependent parts-whole relationship (A is part of B; if B is deleted, related instances of A are also deleted)
Shared aggregation		Parts-whole relationship (A is part of B; if B is deleted, related instances of A need not be deleted)
Generalization		Inheritance relationship (A inherits from B)
Object		Instance of a class
Link		Relationship between objects

Name	*Notation*	*Description*
Lifeline		Interaction partners involved in the communication
Destruction event		Time at which an interaction partner ceases to exist
Combined fragment		Control constructs
Synchronous message		Sender waits for a response message
Response message		Response to a synchronous message
Asynchronous message		Sender continues its own work after sending the asynchronous message
Lost message		Message to an unknown receiver
Found message		Message from an unknown sender

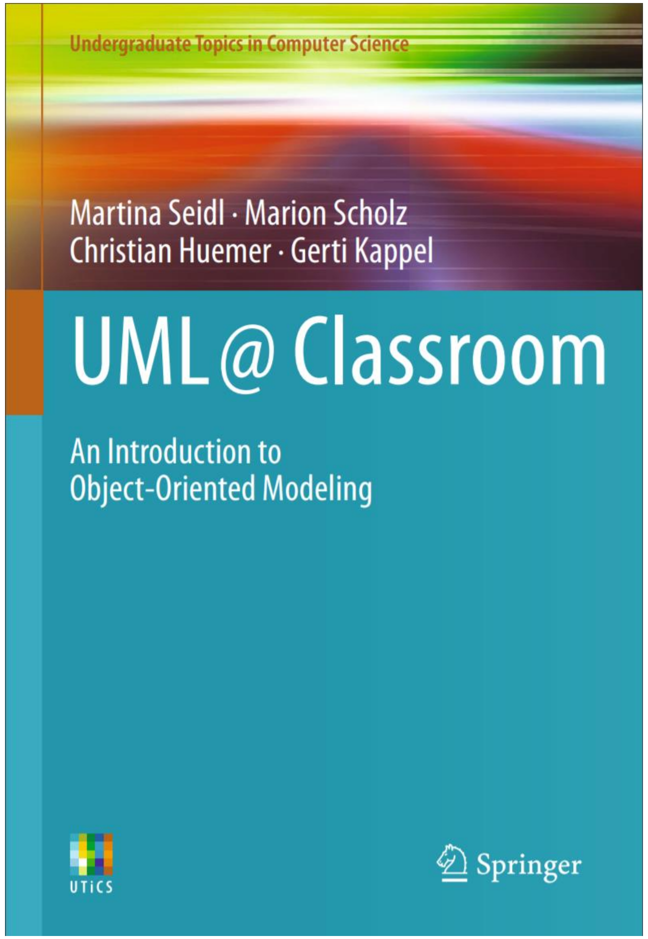
Name	*Notation*	*Description*
Action node		Actions are atomic, i.e., they cannot be broken down further
Activity node		Activities can be broken down further
Initial node		Start of the execution of an activity
Activity final node		End of ALL execution paths of an activity
Flow final node		End of ONE execution path of an activity
Decision node		Splitting of one execution path into two or more alternative execution paths
Merge node		Merging of two or more alternative execution paths into one execution path
Parallelization node		Splitting of one execution path into two or more concurrent execution paths
Synchronization node		Merging of two or more concurrent execution paths into one execution path
Edge		Connection between the nodes of an activity
Call behavior action		Action A refers to an activity of the same name
Object node		Contains data and objects that are created, changed, and read
Parameters for activities		Contain data and objects as input and output parameters
Parameters for actions (pins)		Contain data and objects as input and output parameters

Name	*Notation*	*Description*
Partition		Grouping of nodes and edges within an activity
Send signal action		Transmission of a signal to a receiver
Asynchronous accept (time) event action		Wait for an event E or a time event T
Exception handler		Exception handler is executed instead of the action in the event of an error e
Interruptible activity region		Flow continues on a different path if event E is detected

Name	*Notation*	*Description*
State	 S entry/Activity(...) do/Activity(...) exit/Activity(...)	Description of a specific "time span" in which an object finds itself during its "life cycle". Within a state, activities can be executed on the object.
Transition	 S $\xrightarrow{e}$ T	State transition e from a source state S to a target state T
Initial state		Start of a state machine diagram
Final state		End of a state machine diagram
Terminate node		Termination of an object's state machine diagram
Decision node		Node from which multiple alternative transitions can proceed
Parallelization node		Splitting of a transition into multiple parallel transitions
Synchronization node		Merging of multiple parallel transitions into one transition
Shallow and deep history state	 H / H*	"Return address" to a substate or a nested substate of a composite state

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

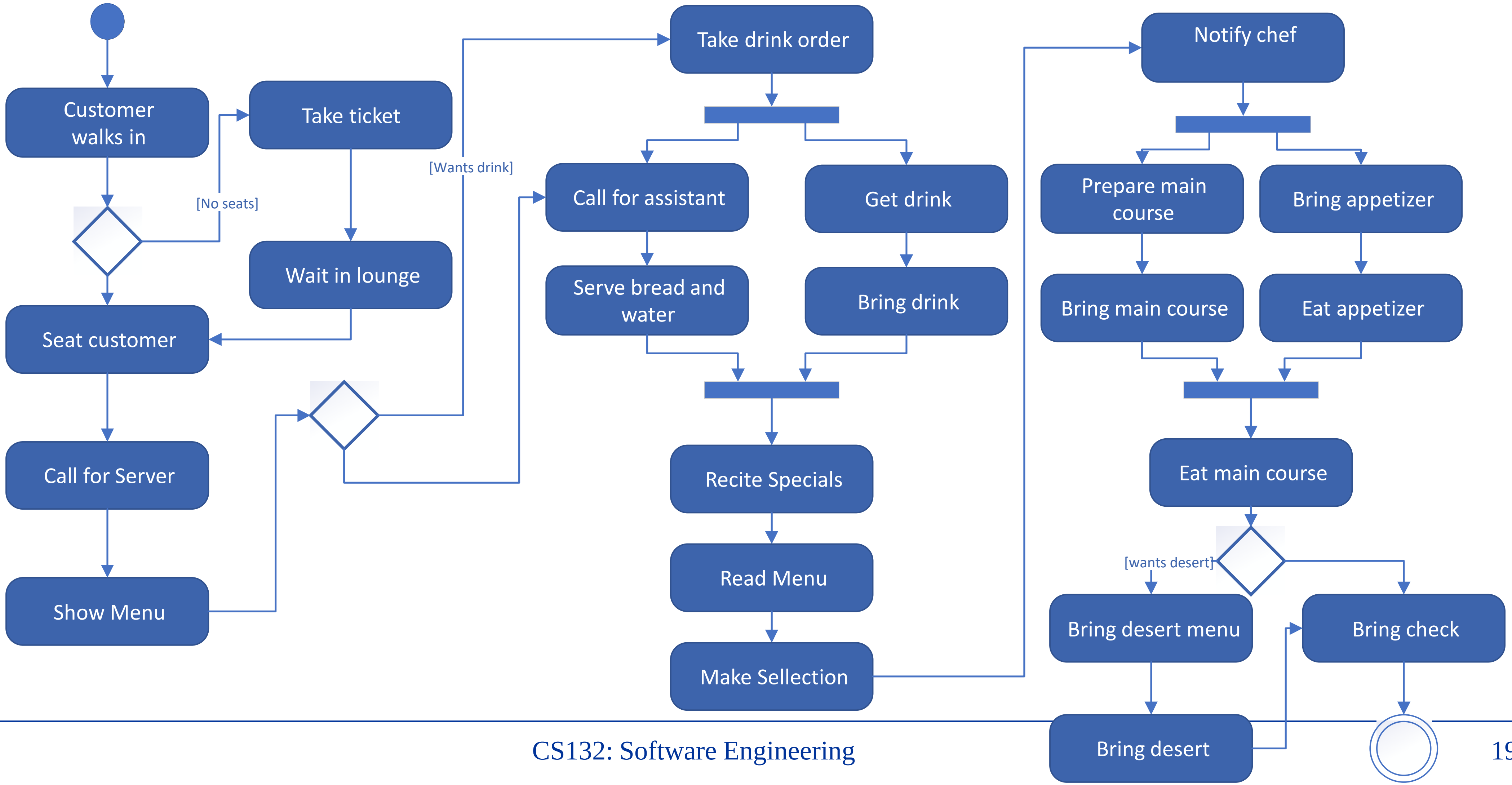
- The owner of restaurant A would like to improve service efficiency

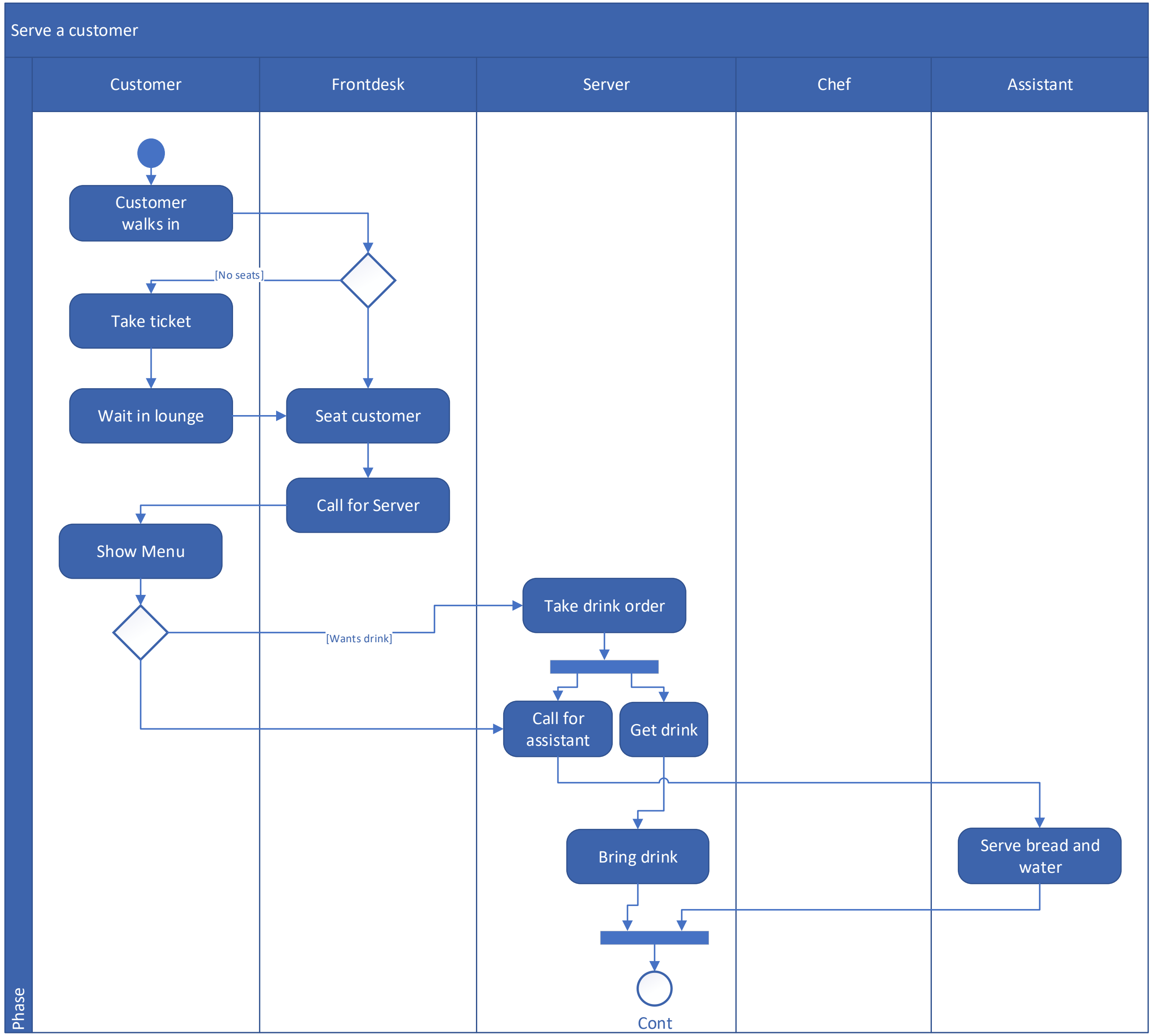
Customer

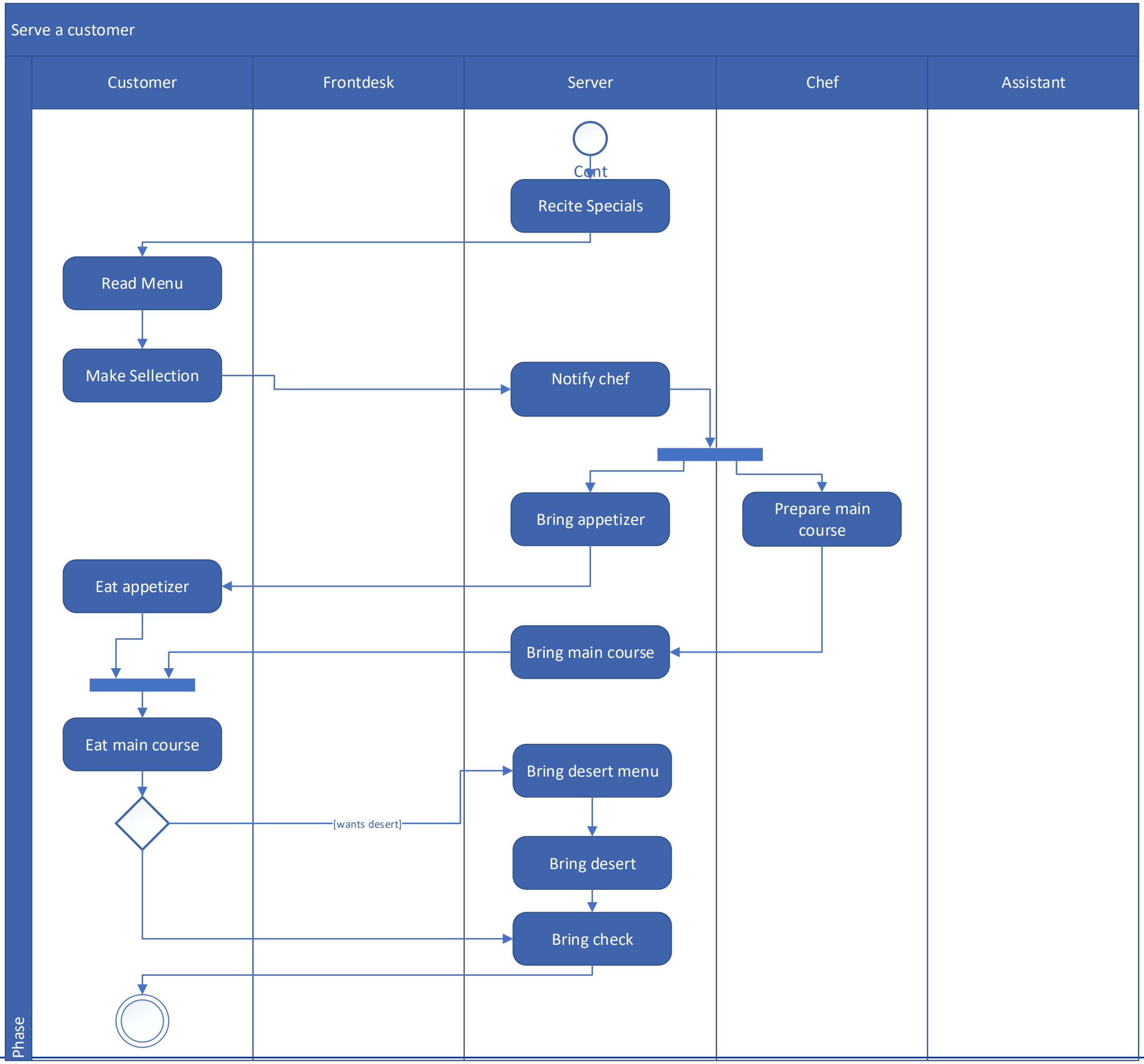
Server

Chef

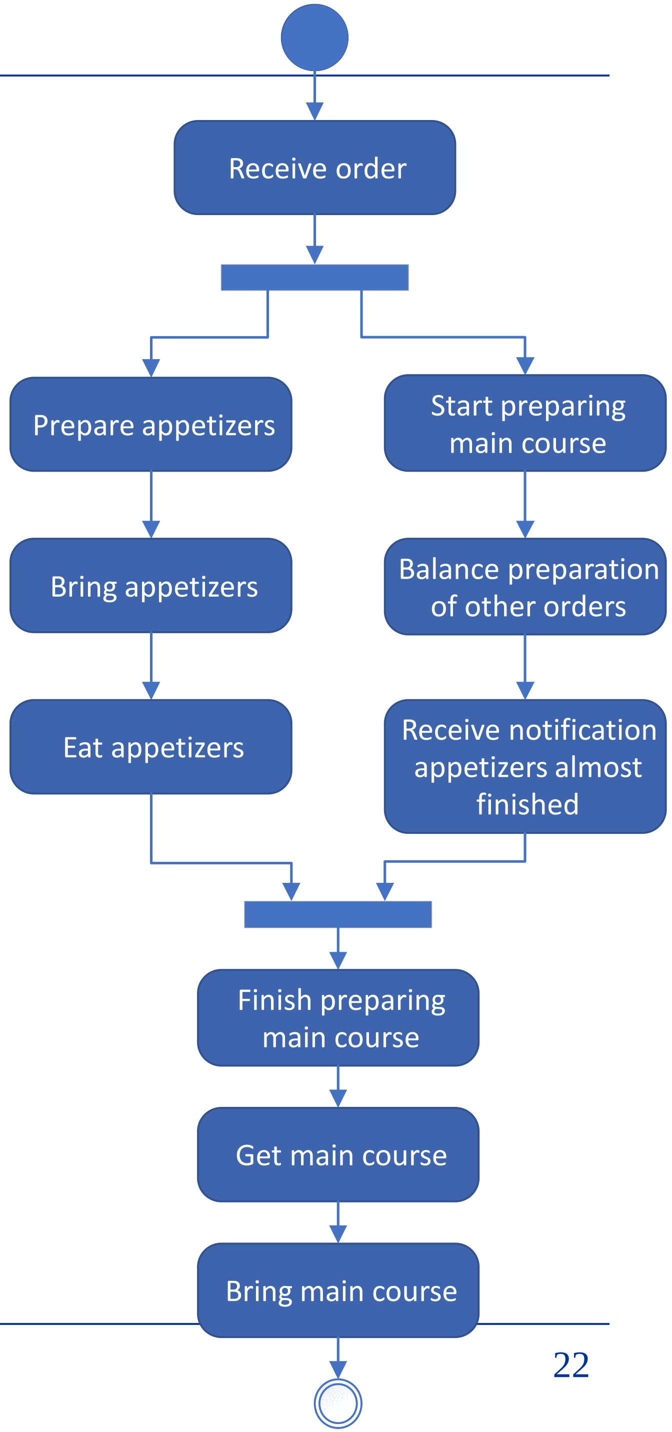
Discover Domain Procedures

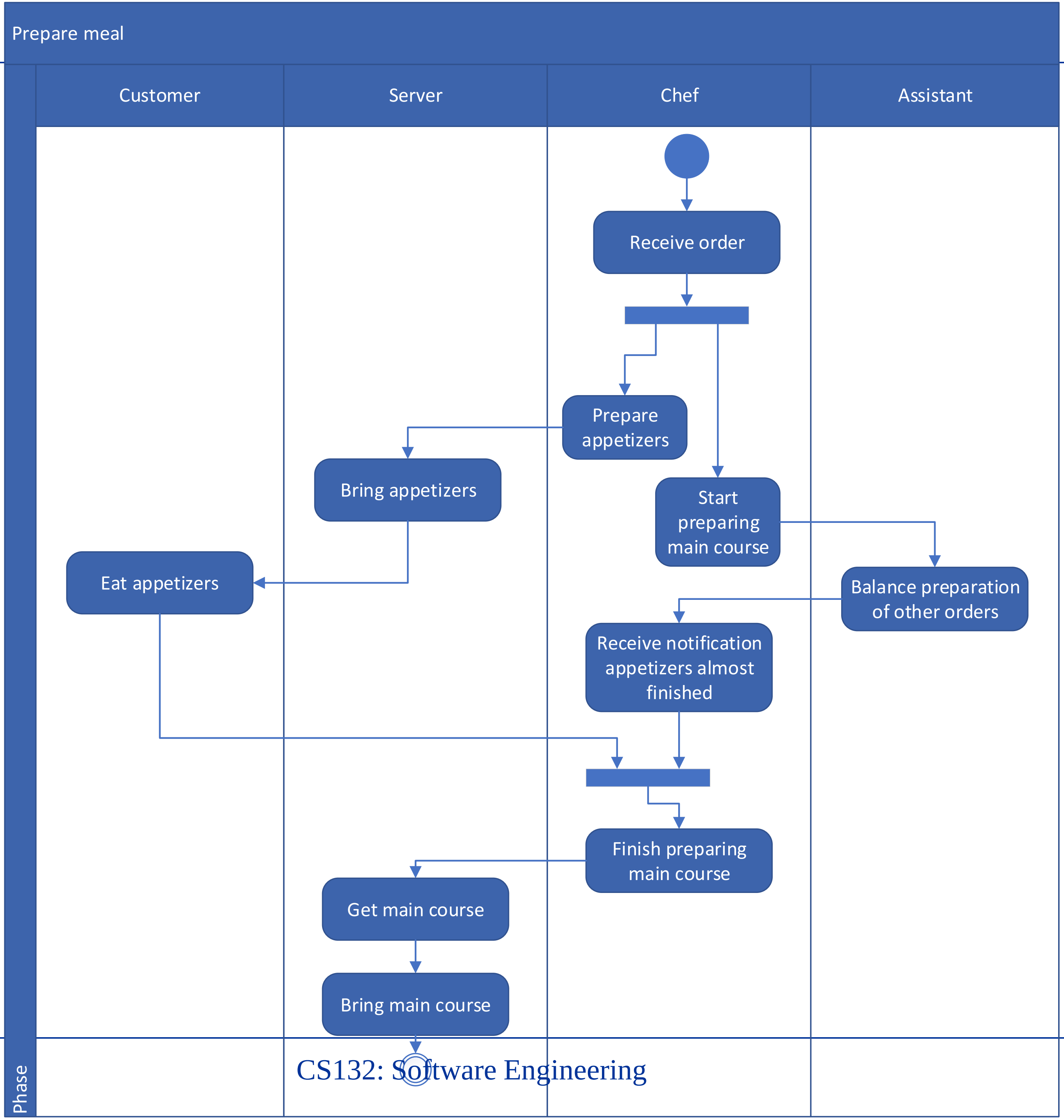



Serve a customer
Customer
Frontdesk
Server
Chef
Assistant
Phase
Customer walks in
[No seats]
Take ticket
Wait in lounge
Seat customer
Call for Server
Show Menu
[Wants drink]
Take drink order
Call for assistant
Get drink
Bring drink
Serve bread and water
Cont

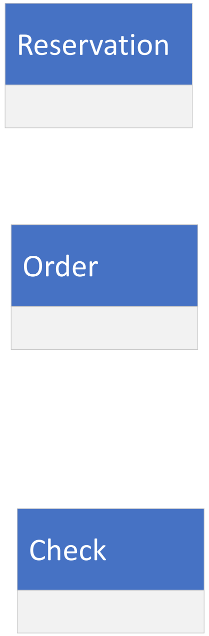

Serve a customer
Customer
Frontdesk
Server
Chef
Assistant
Phase
Cont
Recite Specials
Read Menu
Make Sellection
Notify chef
Bring appetizer
Prepare main course
Eat appetizer
Bring main course
Eat main course
Bring desert menu
[wants desert]
Bring desert
Bring check

- Prepare food

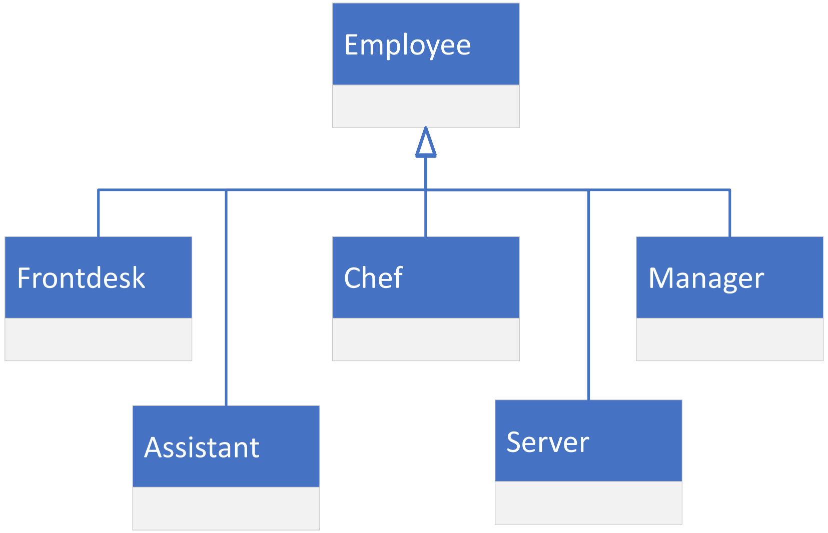


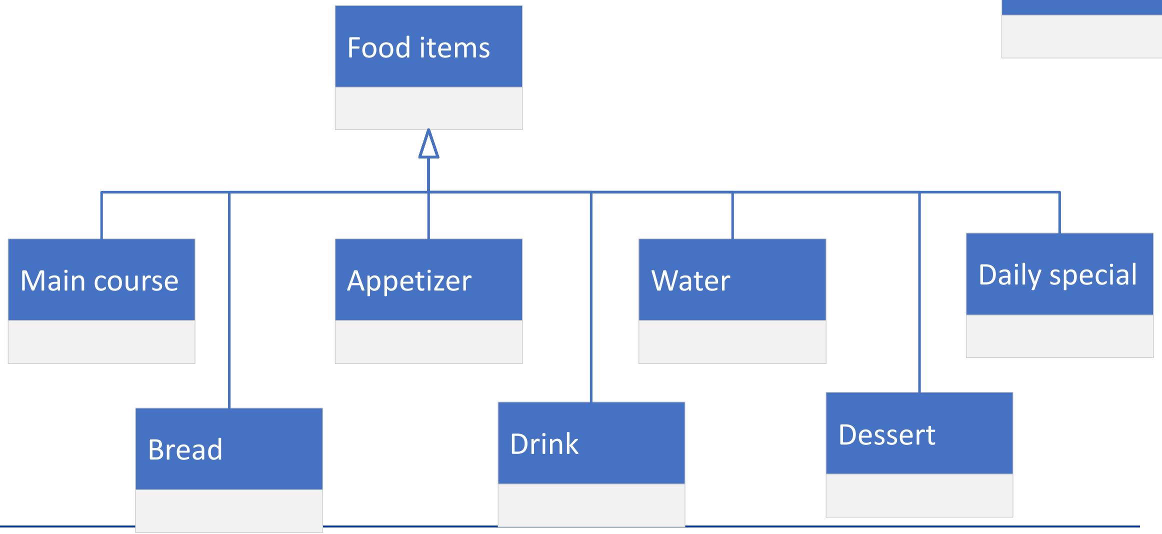

Prepare meal
Customer
Server
Chef
Assistant
Receive order
Prepare appetizers
Bring appetizers
Start preparing main course
Balance preparation of other orders
Eat appetizers
Receive notification appetizers almost finished
Finish preparing main course
Get main course
Bring main course
Phase

Domain Analysis



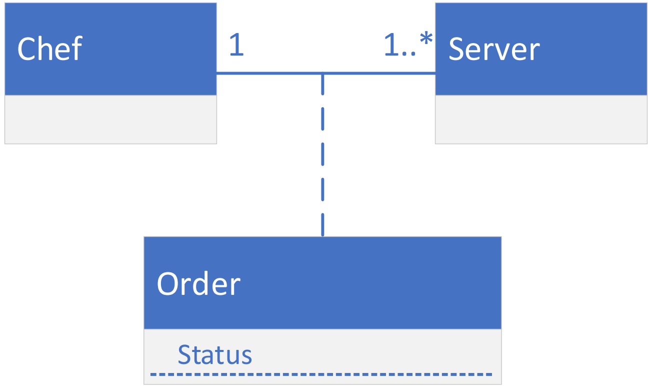
1. Develop 1st version of class diagram
2. Find similar attributes and organize objects into classes

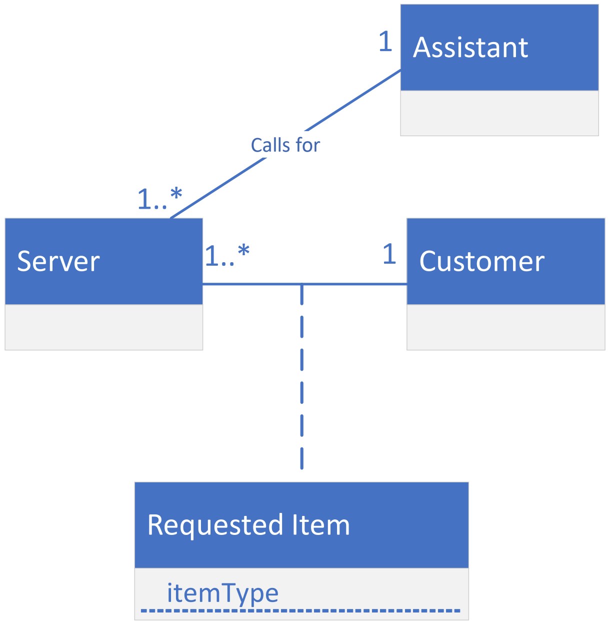


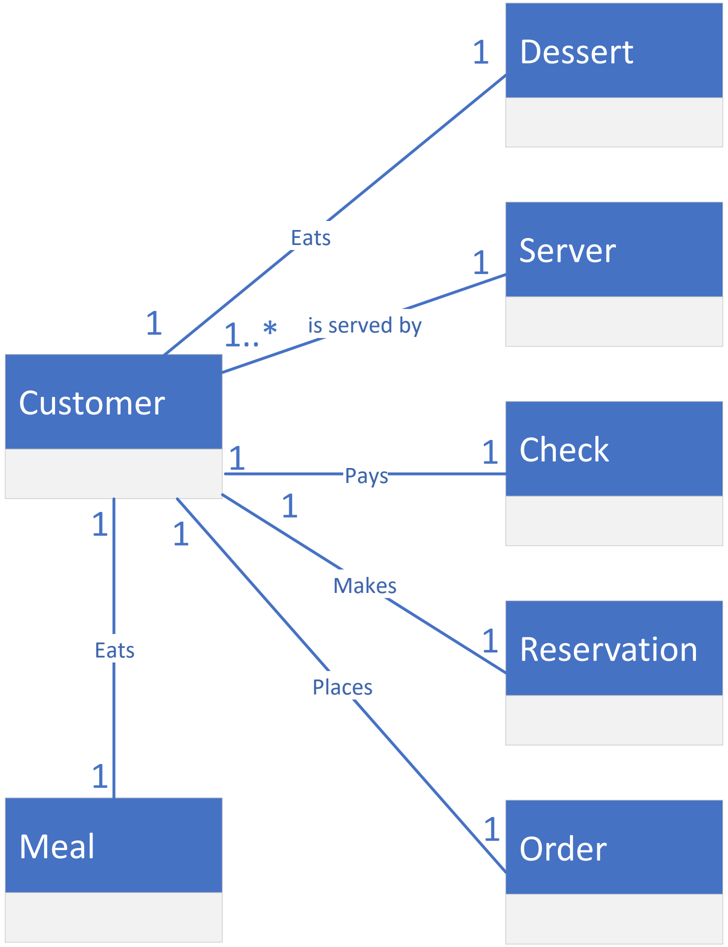


Domain Analysis (cont.)

3. Further understand the domain
   – Find associations

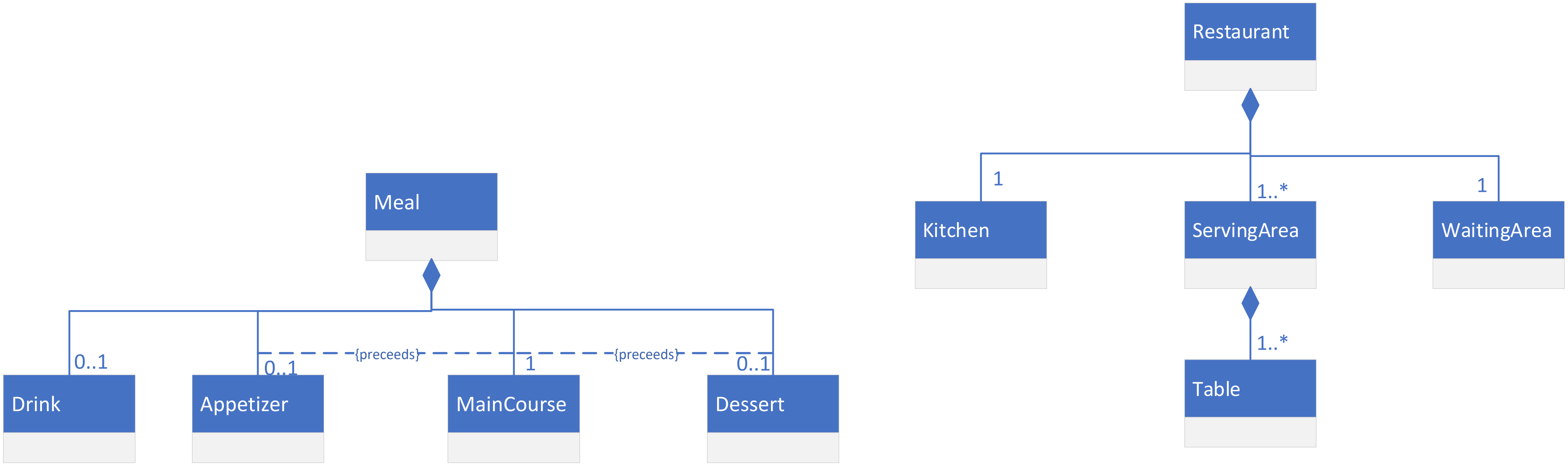






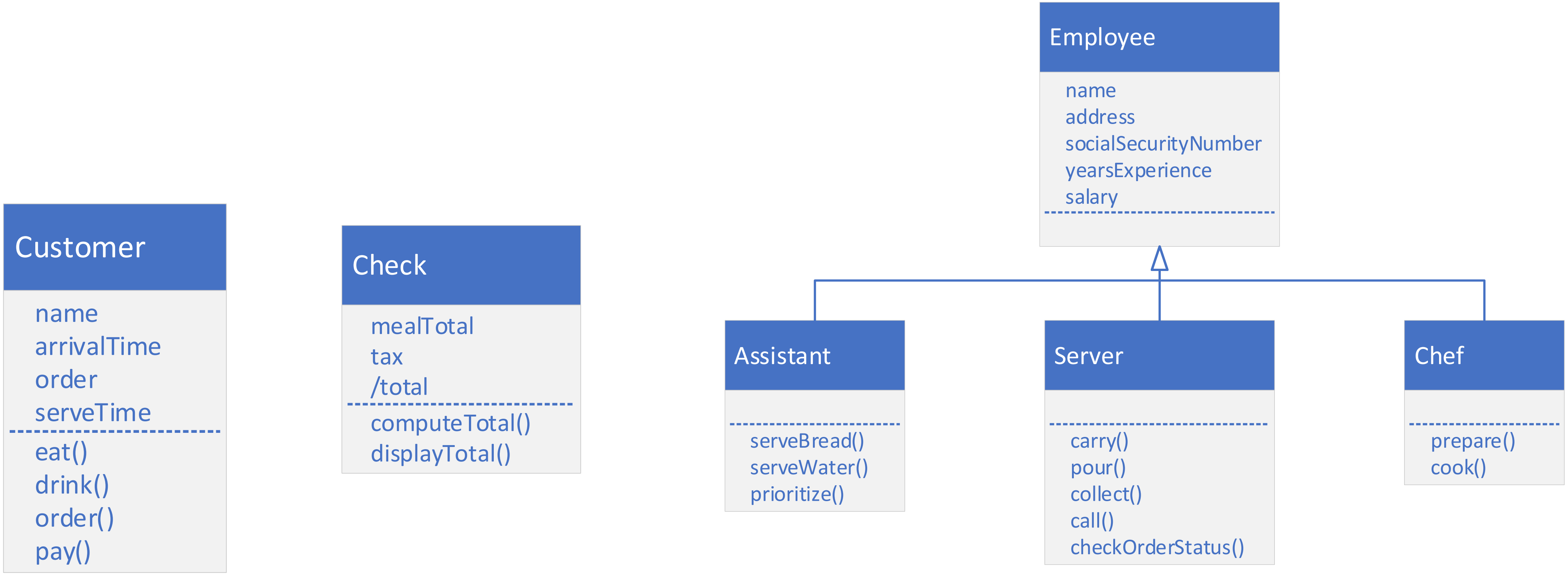
Domain Analysis (cont.)

4. Find aggregations and compositions



Domain Analysis (cont.)

5. Enrich information in classes

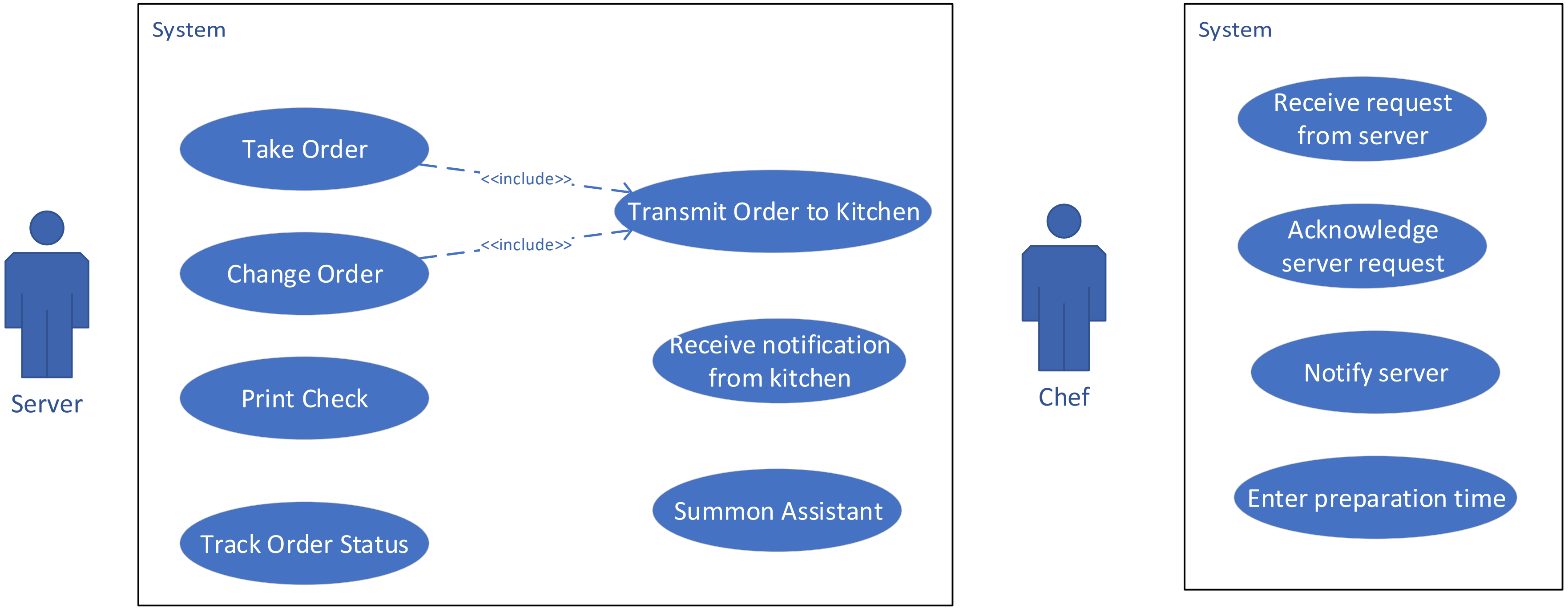


Discover system requirements

- 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

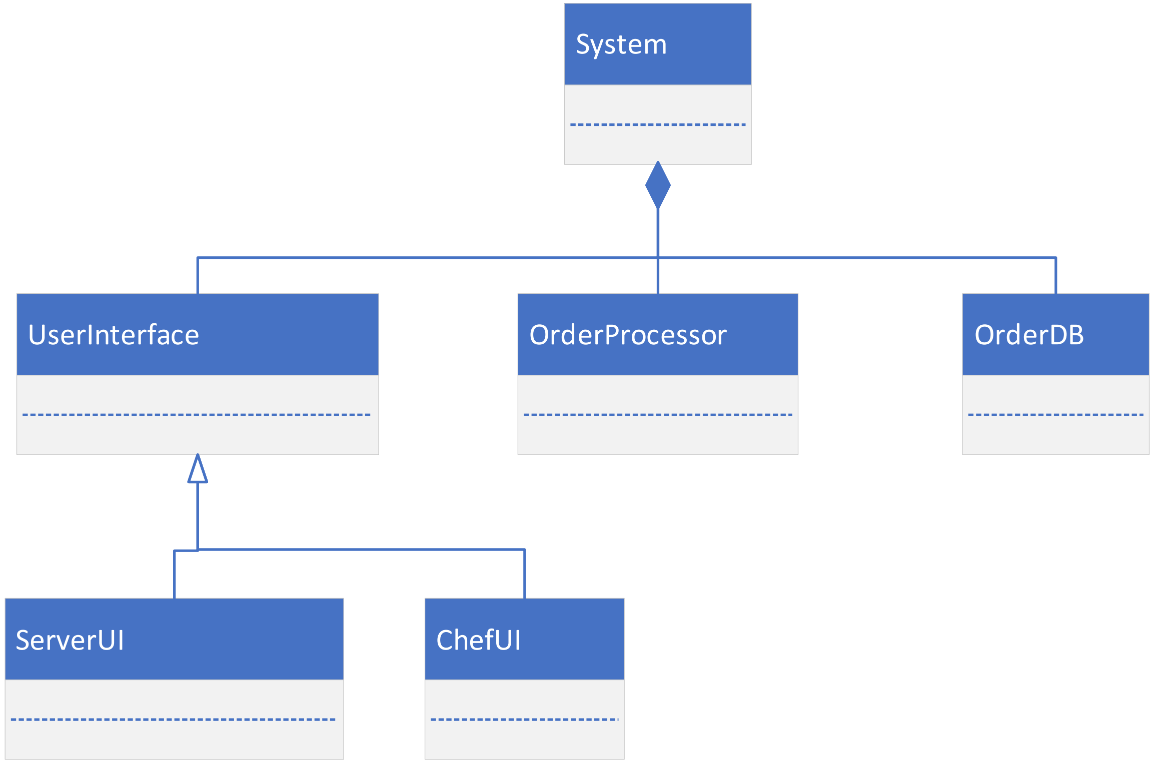
Discover system requirements (cont.)

- System requirements as use cases



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



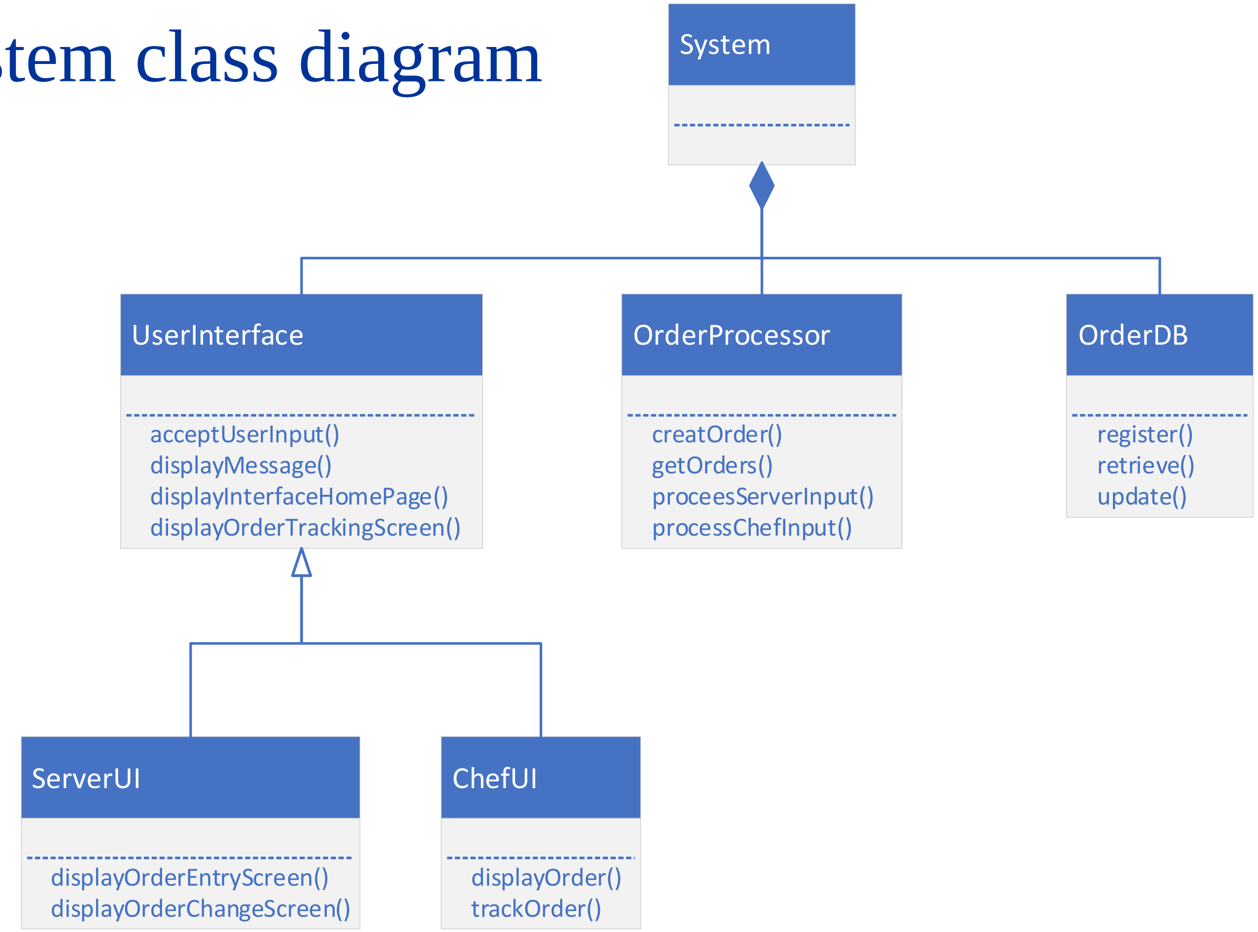
- Proposed solution
  - An order database that keeps track of order information
  - An order processor that handles order generation/modification
  - User interface for both the chef and the server

Discover system requirements (cont.)

- 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

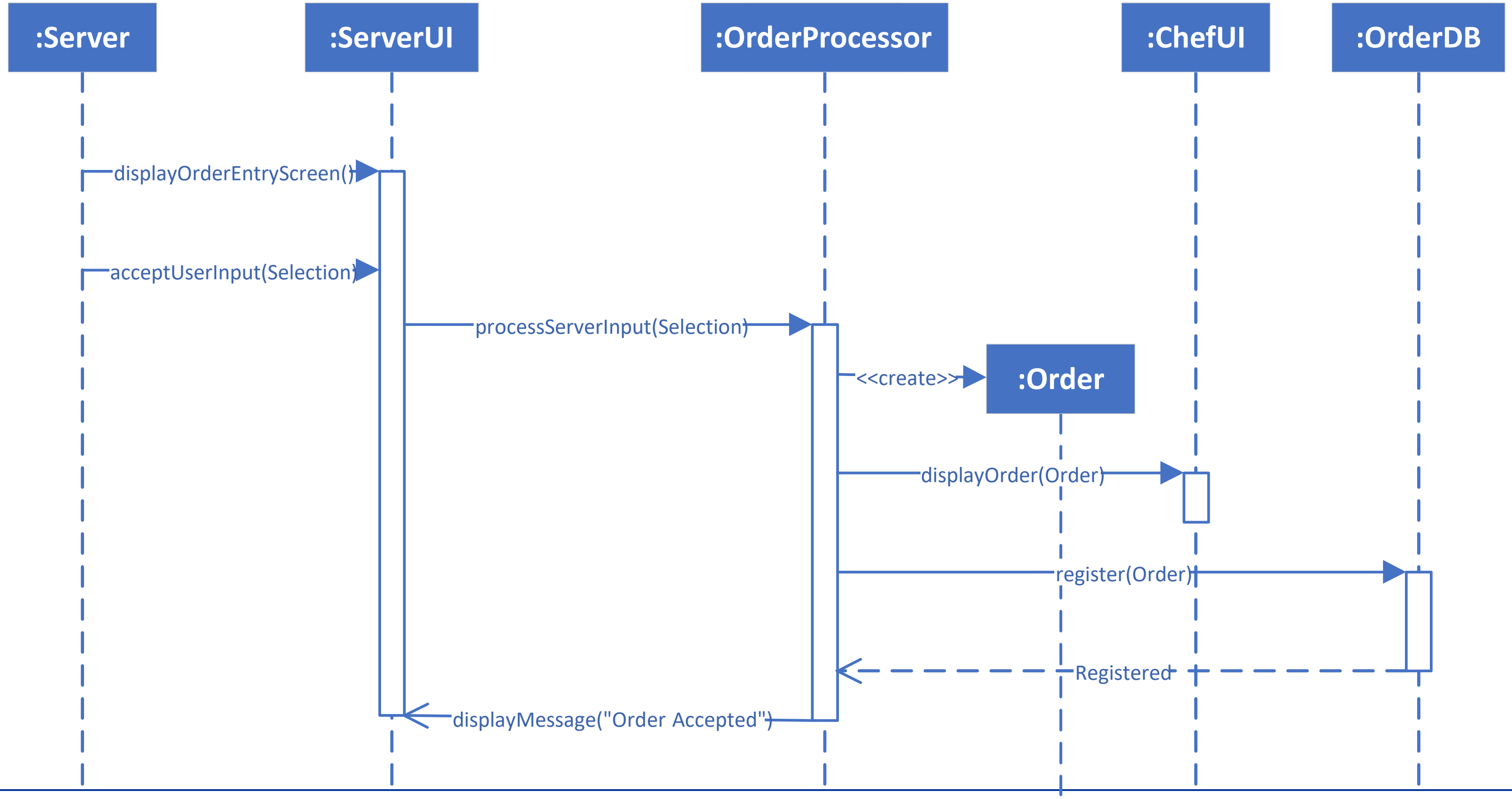
Discover system requirements (cont.)

- Enriched system class diagram



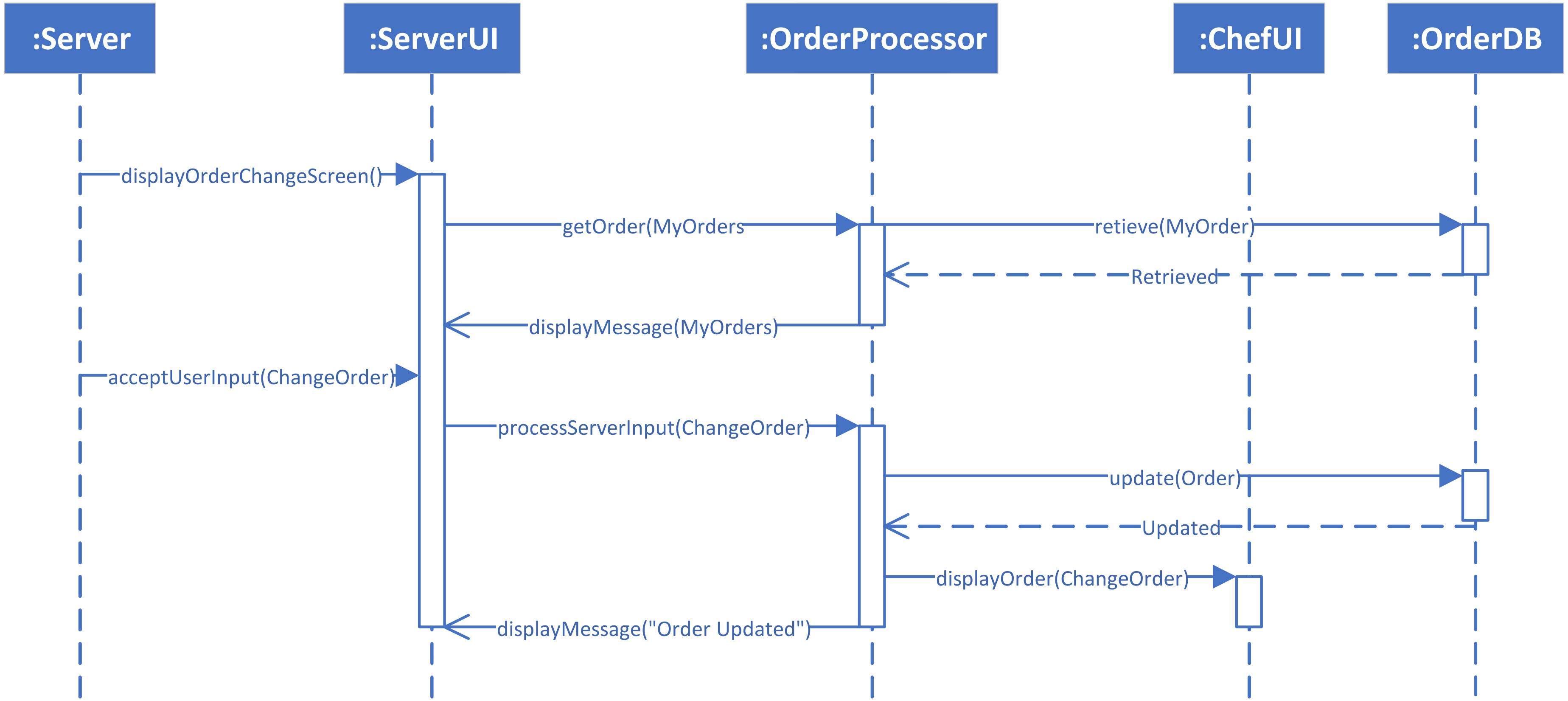
Identify interactions

- Use case “Take an order”



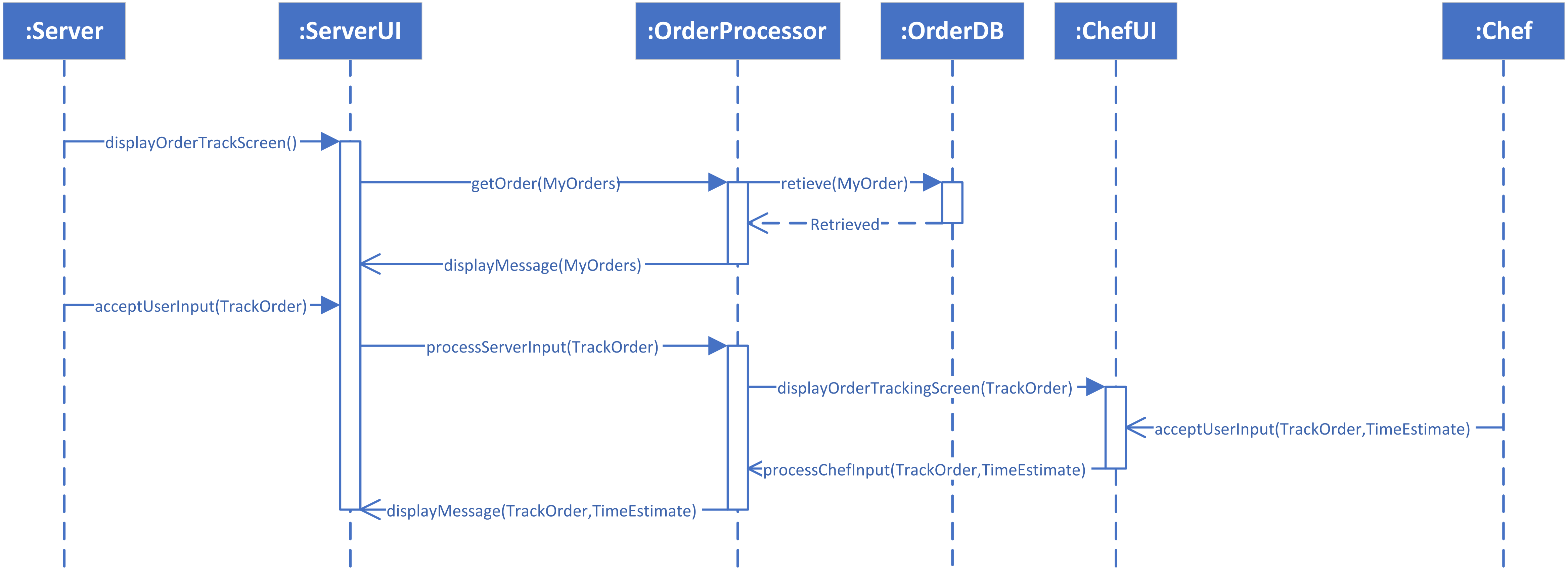
Identify interactions (cont.)

- 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

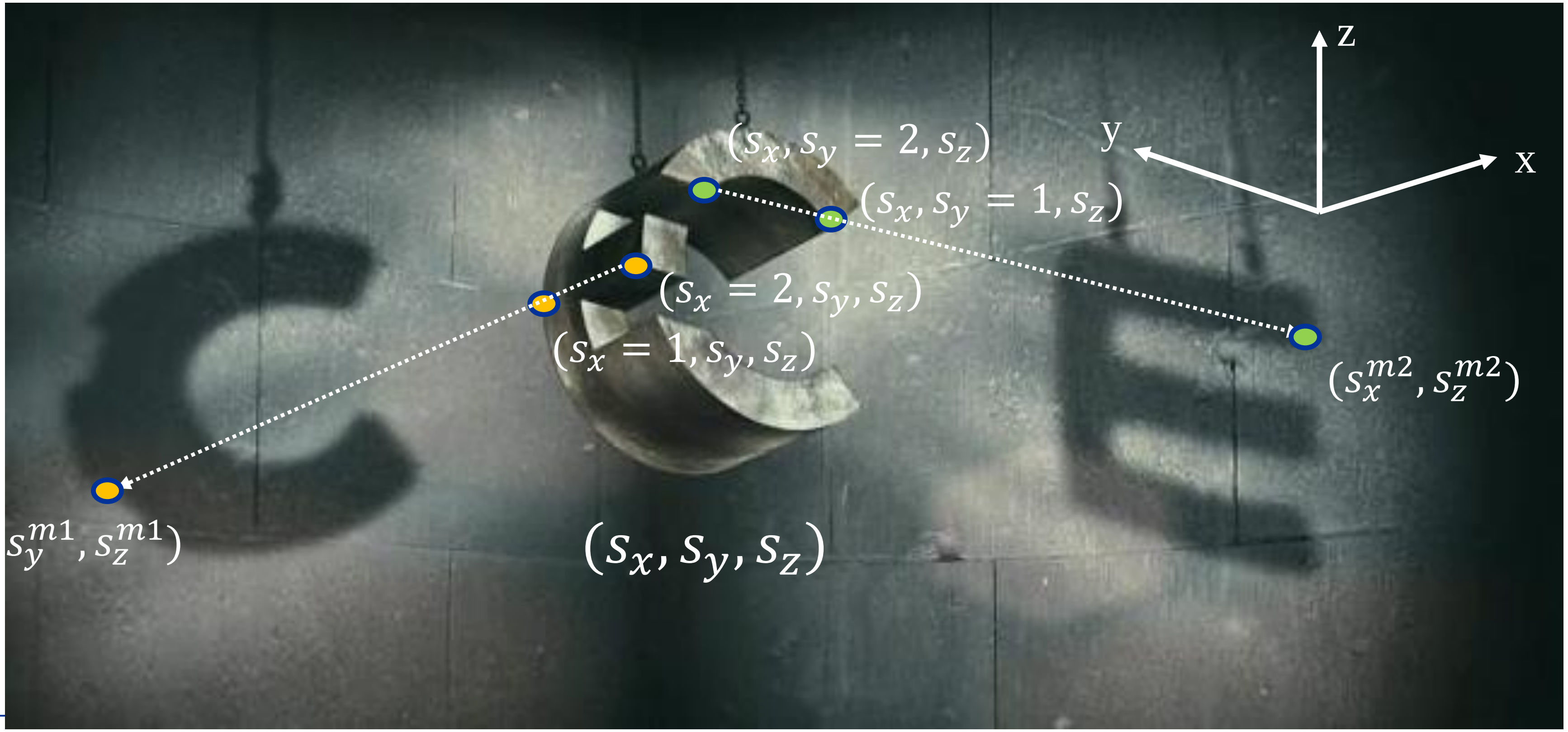
What are models?

- 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

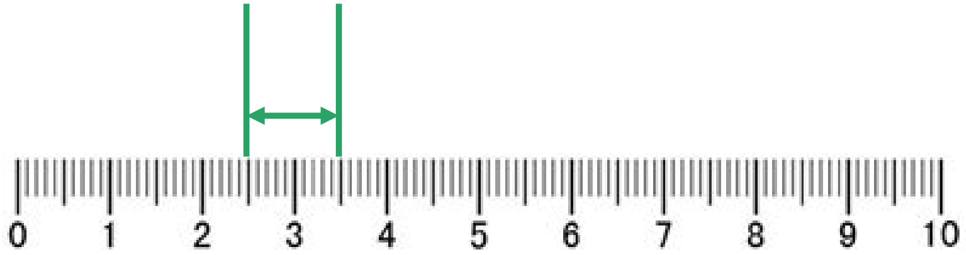
- States (s_x, s_y, s_z) *are abstracted to* (s_y^{m1}, s_z^{m1})
  - $(s_x, s_y, s_z) \rightarrow (s_y^{m1}, s_z^{m1})$
- Loss of information



Abstraction: Approximation of state variable values

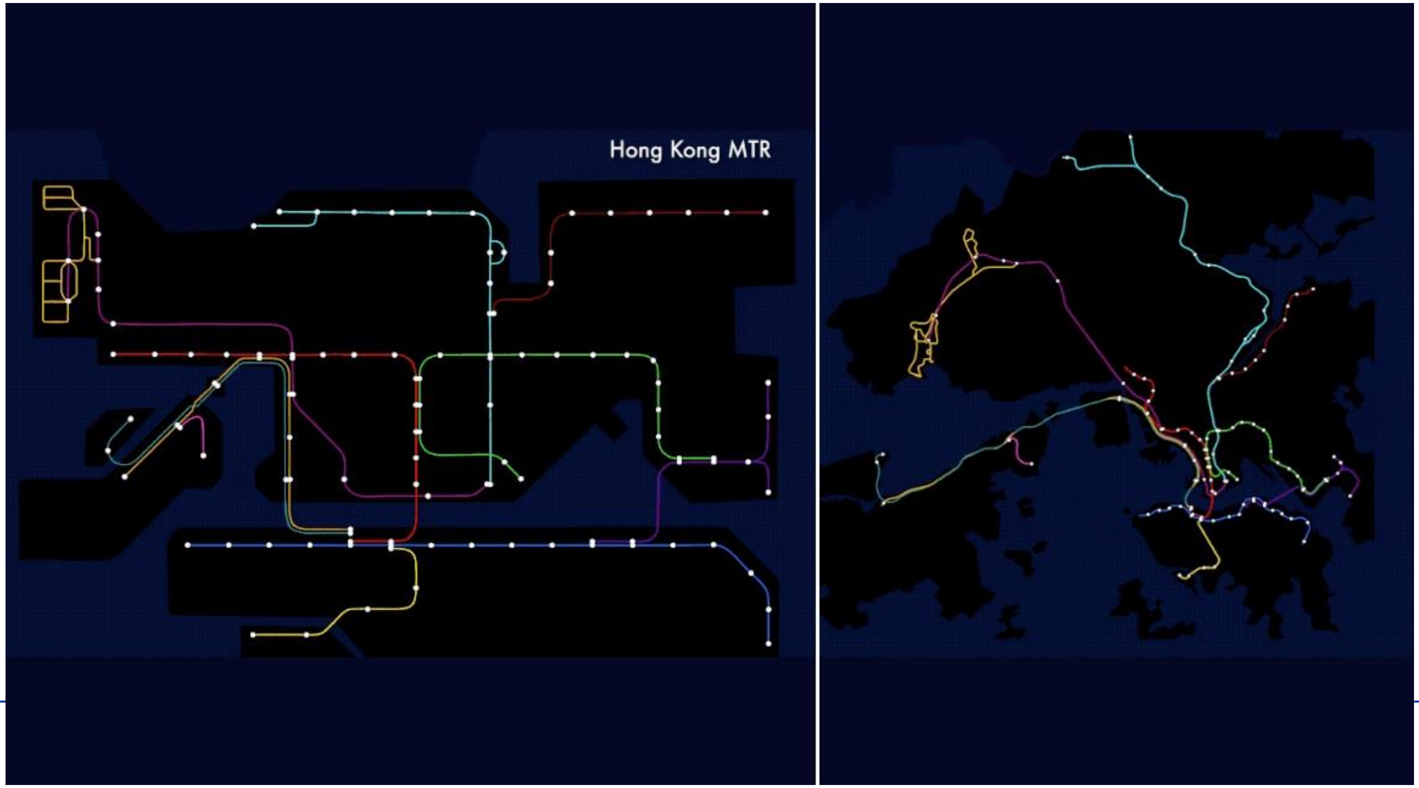
- Irrational numbers
  - $\pi \approx 3.1415$
  - $\sqrt{2} \approx 1.414$

- Approximation is another way of abstraction

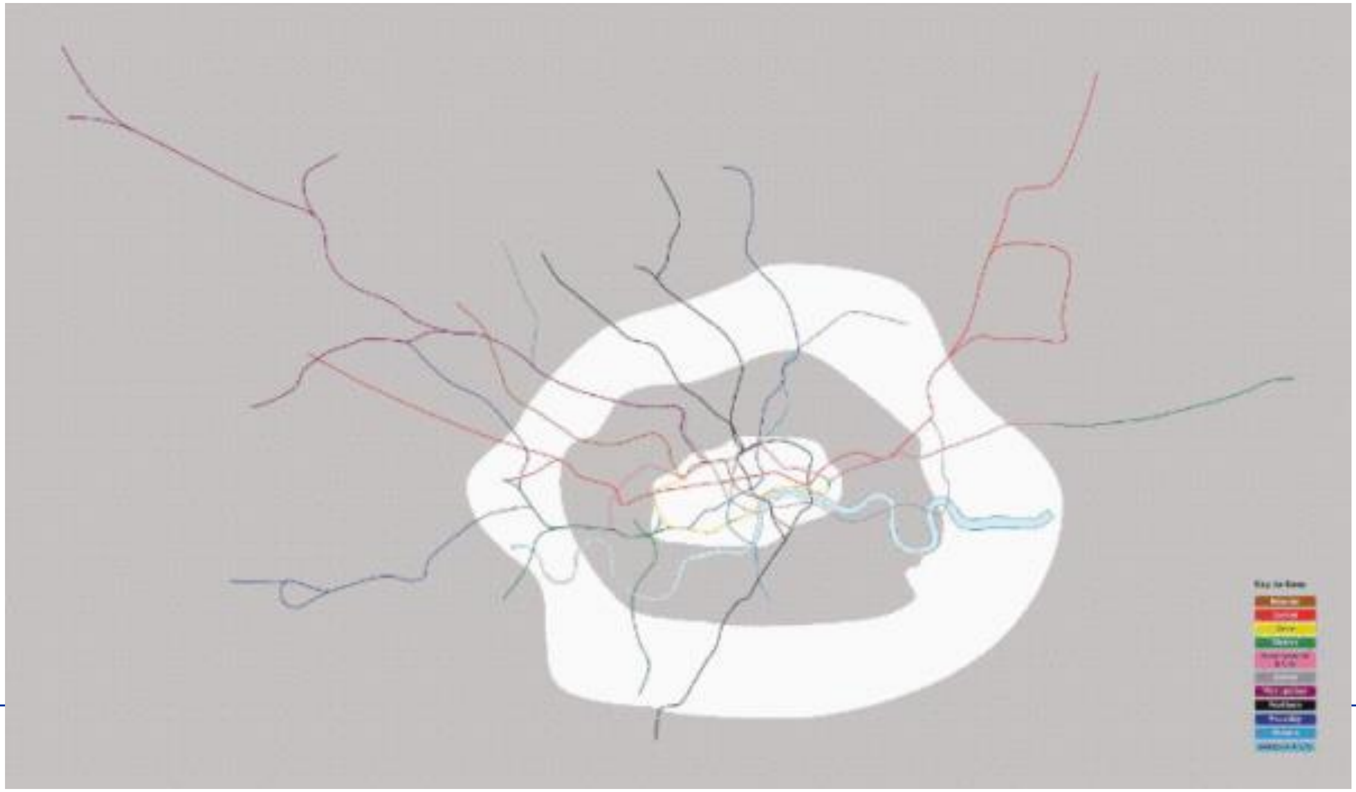


Interpolation: extracting interpretable information

- 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