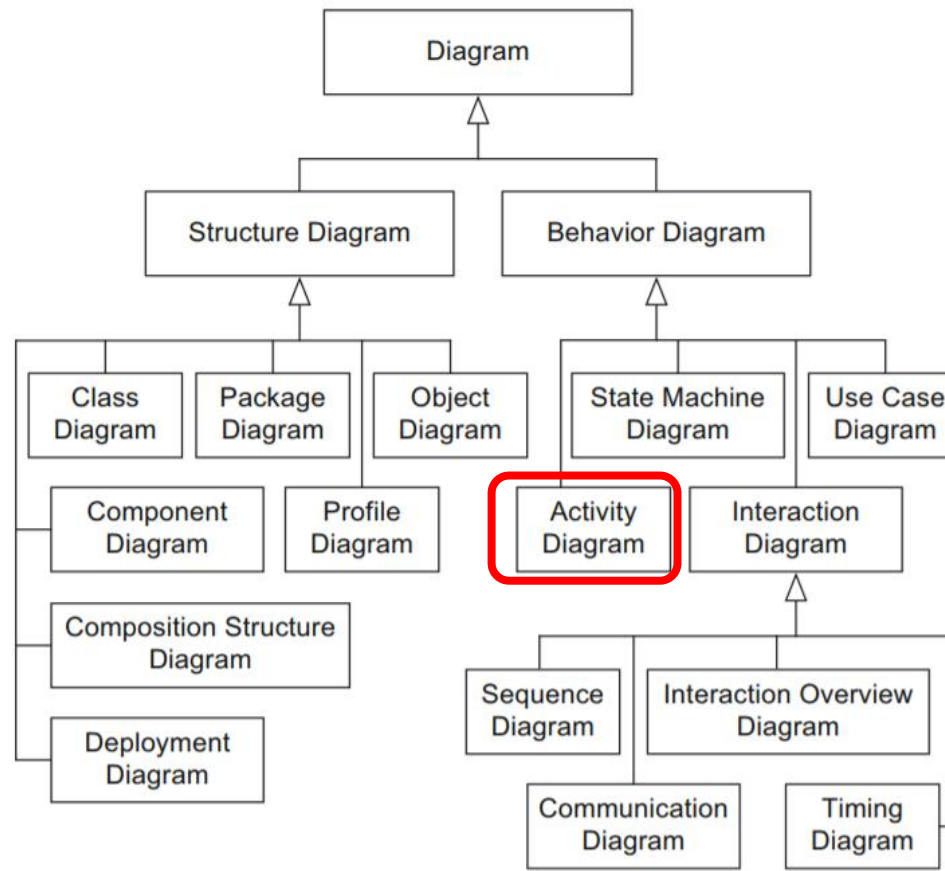






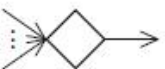
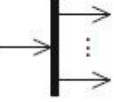
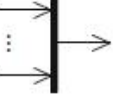

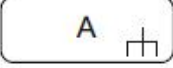
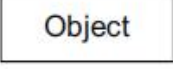


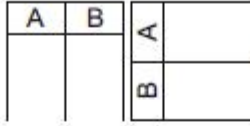

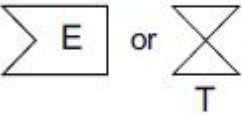
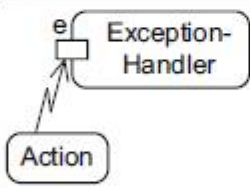
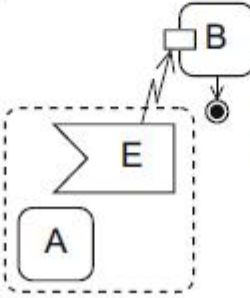


Lecture 7: UML Part 3

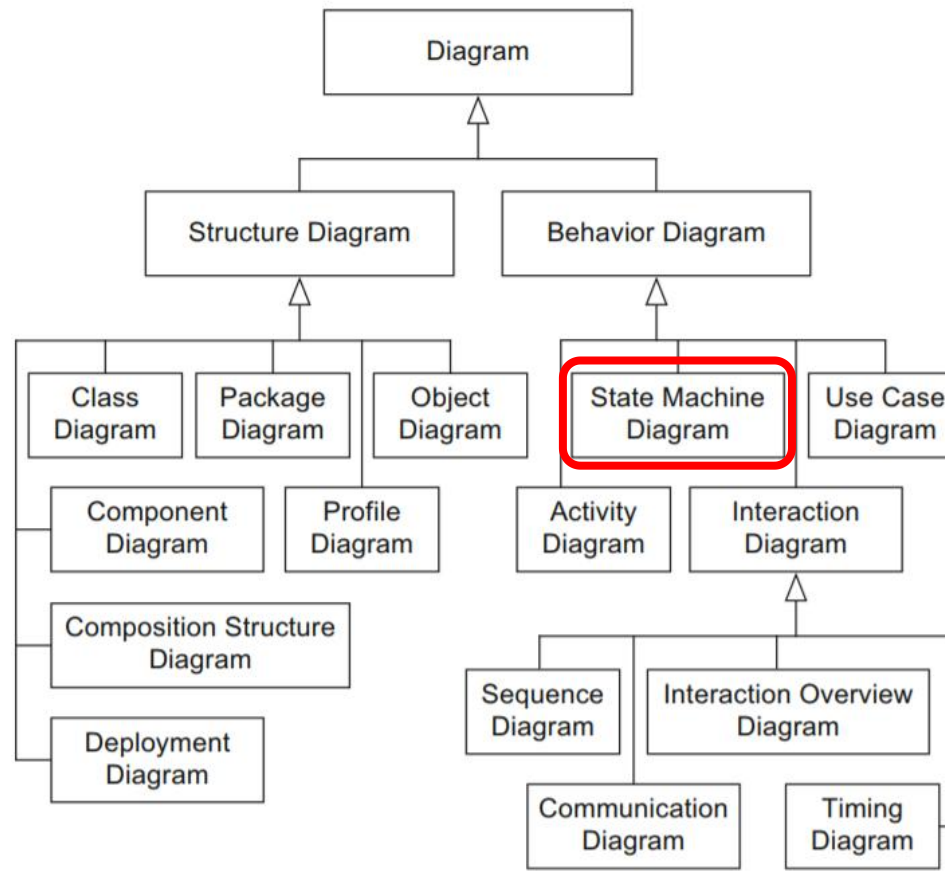
UML Diagrams



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

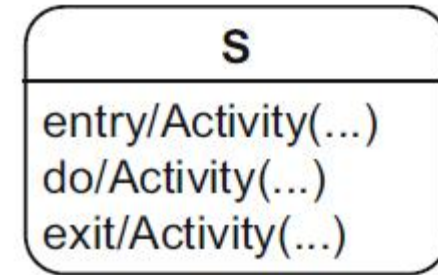
Name	Notation	Description
Partition		Grouping of nodes and edges within an activity
Send signal action		Transmission of a signal to a receiver
Asynchronous accept (time) event action		Wait for an event E or a time event T
Exception handler		Exception handler is executed instead of the action in the event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

UML Diagrams

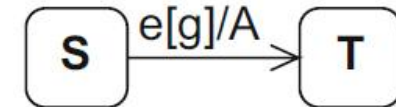


State Machine

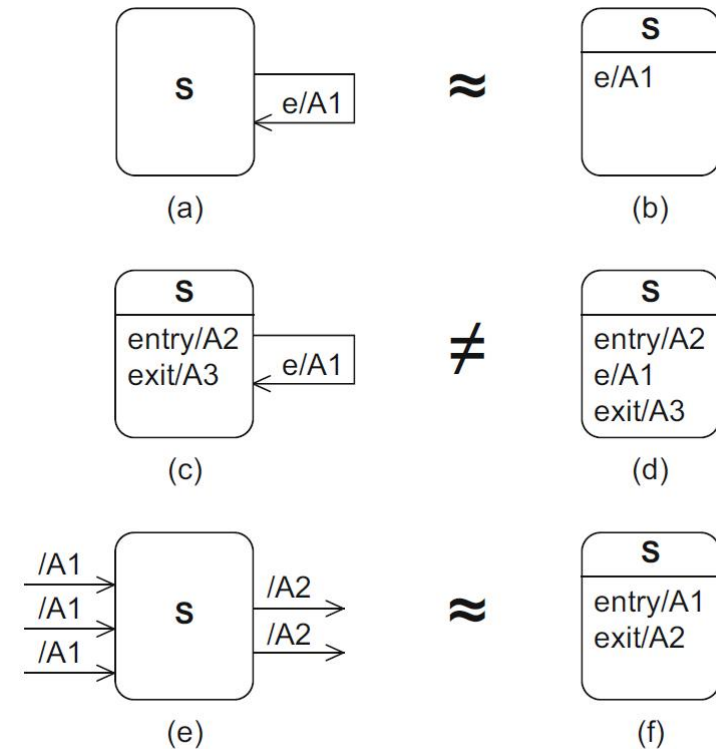
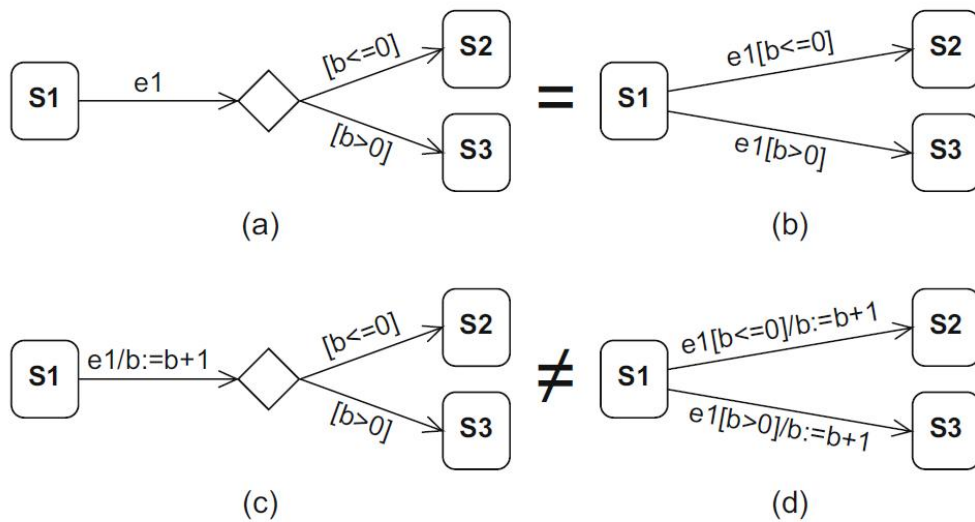
- State
 - entry: upon entering the state
 - do: during the state
 - exit: upon exiting the state
- Transition
 - e: event/trigger for the transition
 - g: guard/condition for the transition
 - A: activity when executing the transition



Transition

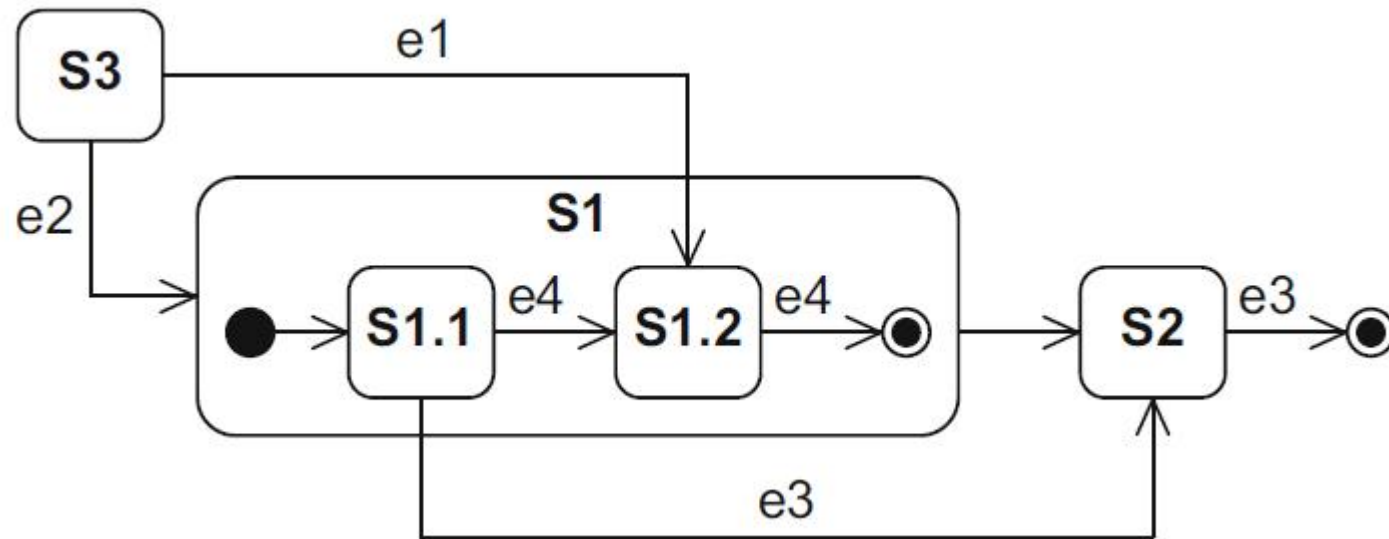


State Machine Diagram: Semantics

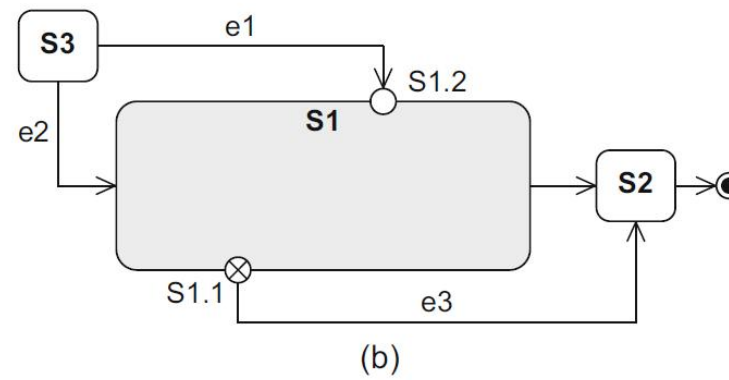
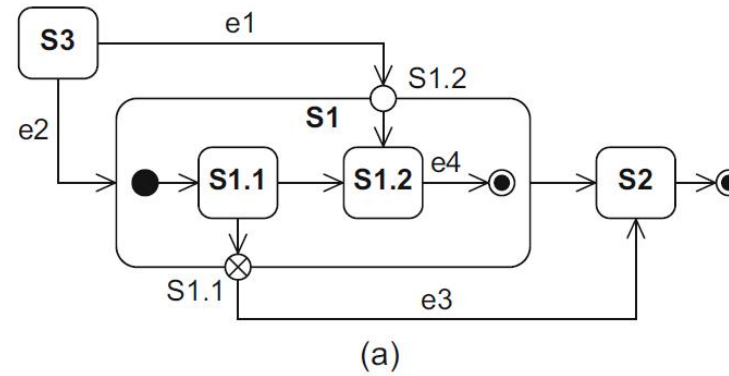


Composite States

- Nested States

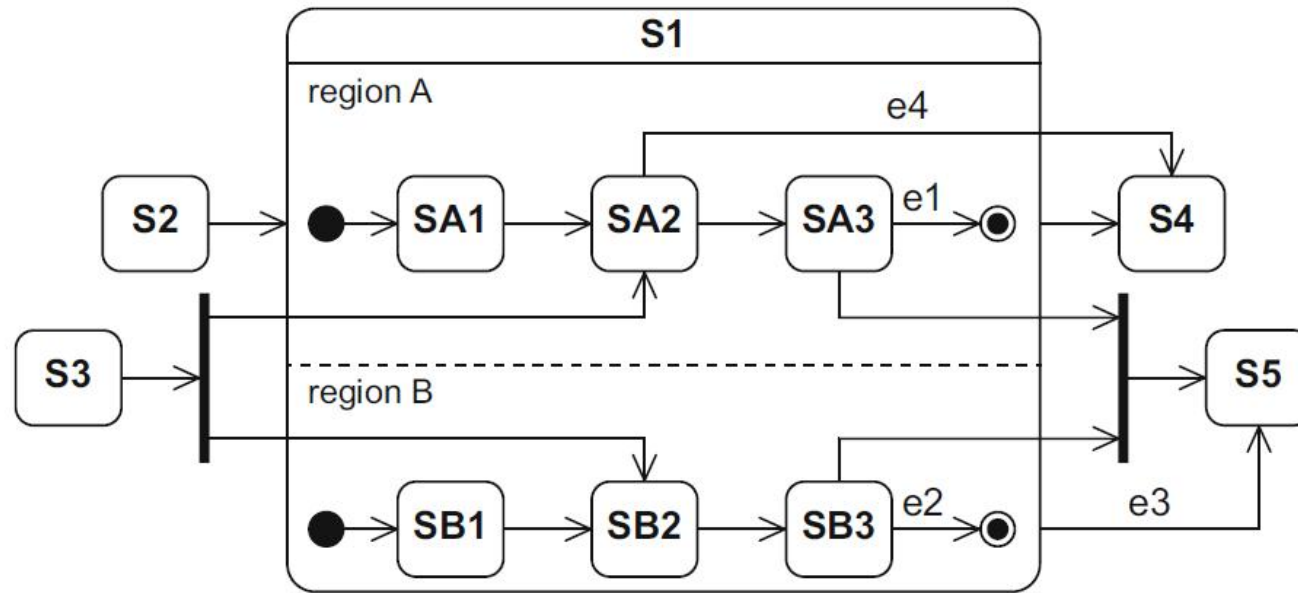


Entry and Exit Points

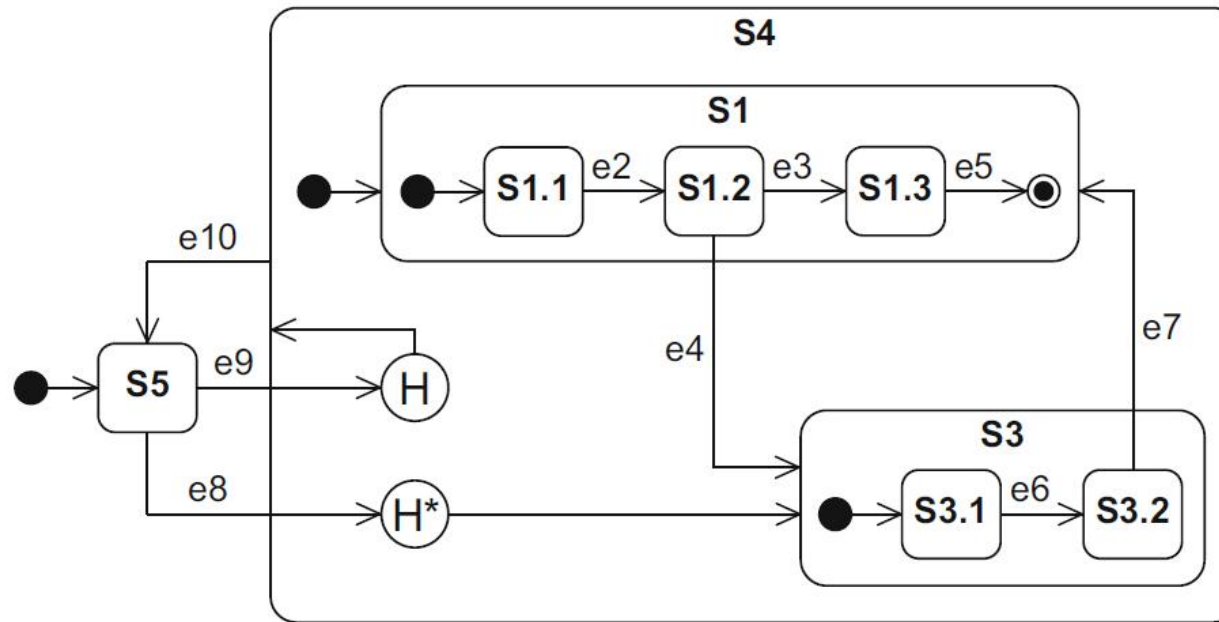


Orthogonal State

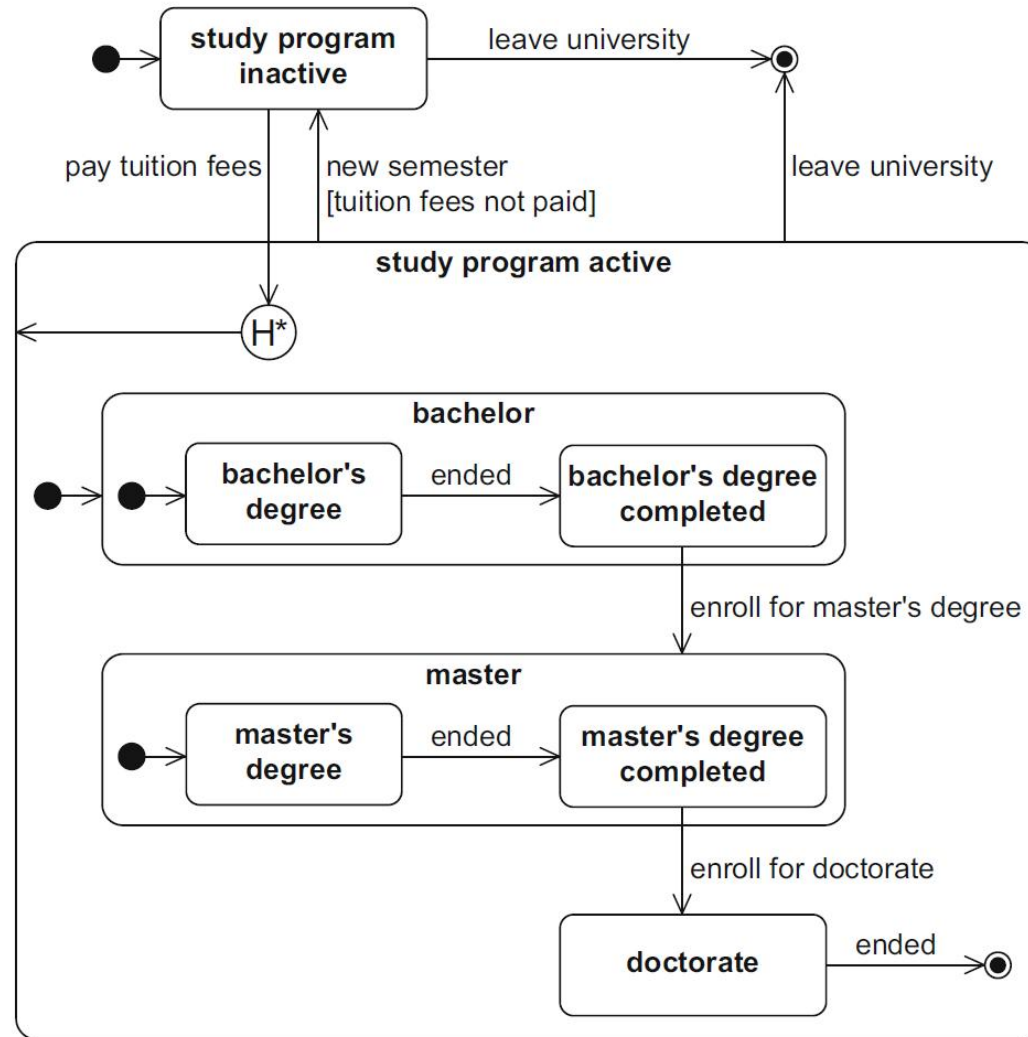
- Parallel State



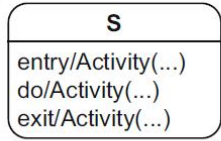
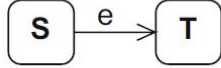



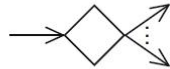
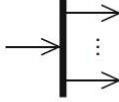
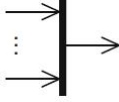

History State



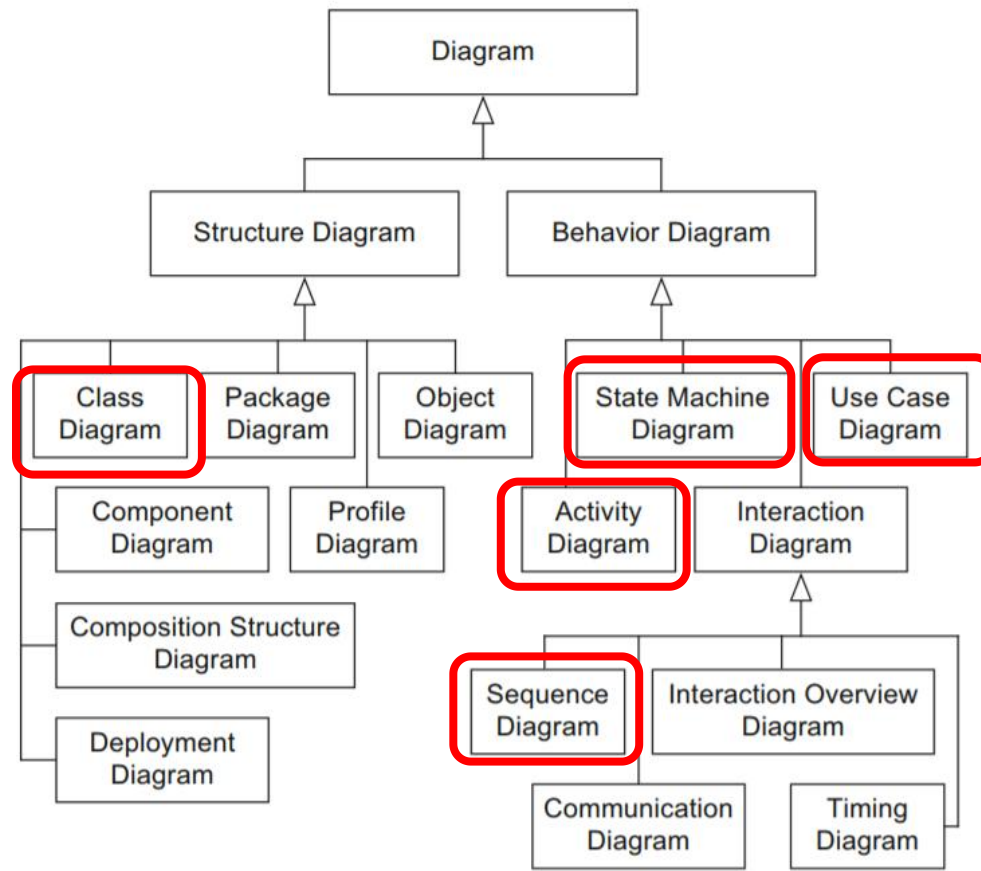
Example



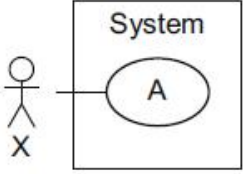
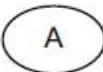
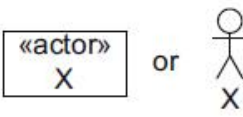
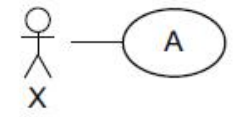
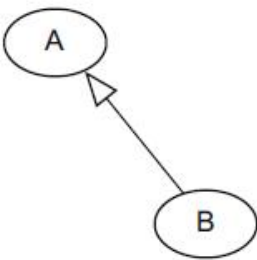
Summary

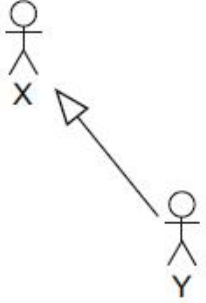
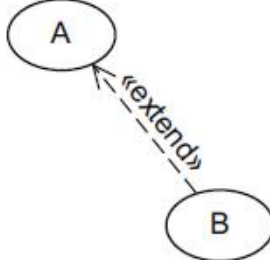
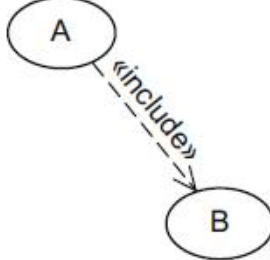
Name	Notation	Description
State		Description of a specific “time span” in which an object finds itself during its “life cycle”. Within a state, activities can be executed on the object.
Transition		State transition e from a source state S to a target state T
Initial state		Start of a state machine diagram
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		“Return address” to a substate or a nested substate of a composite state

UML Diagrams

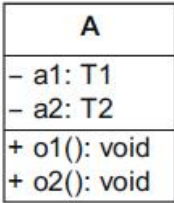

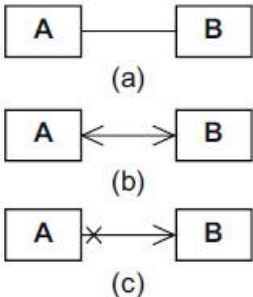
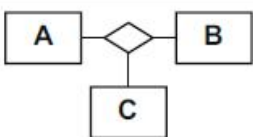


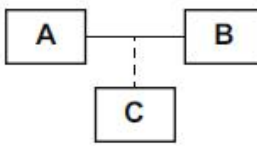
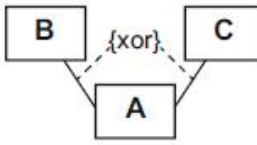
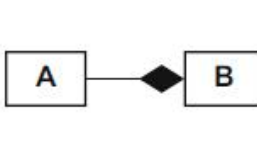
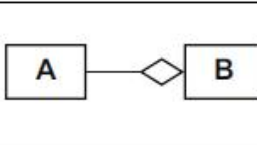
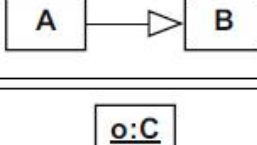


Summary: Use Case Diagram

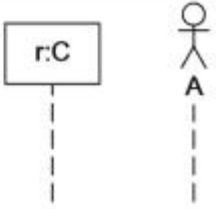
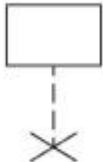
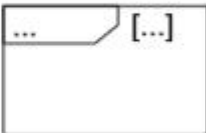



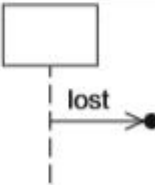

Name	Notation	Description
System		Boundaries between the system and the users of the system
Use case		Unit of functionality of the system
Actor		Role of the users of the system
Association		X participates in the execution of A
Generalization (use case)		B inherits all properties and the entire behavior of A







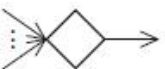
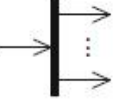
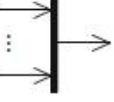

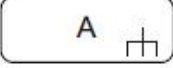
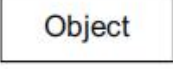


Generalization (actor)		Y inherits from X; Y participates in all use cases in which X participates
Extend relationship		B extends A: optional incorporation of use case B into use case A
Include relationship		A includes B: required incorporation of use case B into use case A

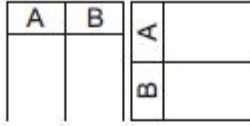

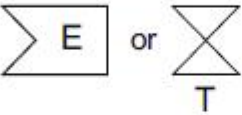
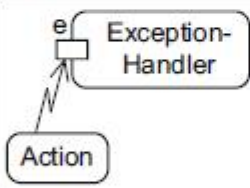
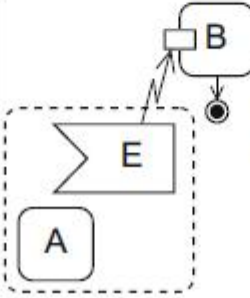
Summary: Class Diagram

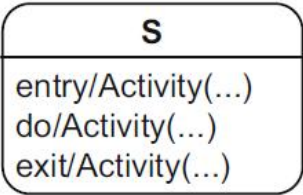
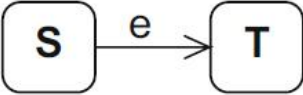



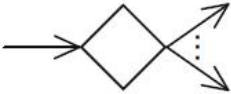
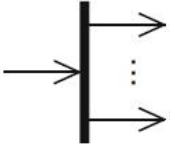
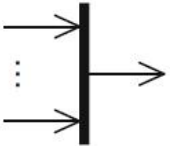
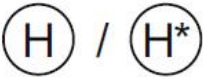
Name	Notation	Description
Class		Description of the structure and behavior of a set of objects
Abstract class		Class that cannot be instantiated
Association		Relationship between classes: navigability unspecified (a), navigable in both directions (b), not navigable in one direction (c)
N-ary association		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

<i>Name</i>	<i>Notation</i>	<i>Description</i>
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

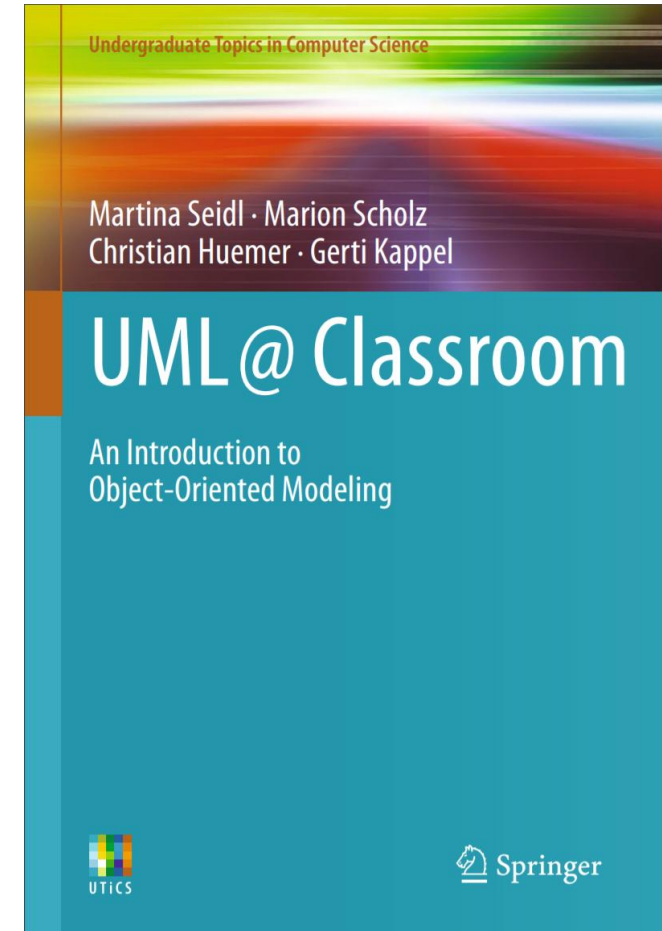
Name	Notation	Description
Action node		Actions are atomic, i.e., they cannot be broken down further
Activity node		Activities can be broken down further
Initial node		Start of the execution of an activity
Activity final node		End of ALL execution paths of an activity
Flow final node		End of ONE execution 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

<i>Name</i>	<i>Notation</i>	<i>Description</i>
State		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		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		“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

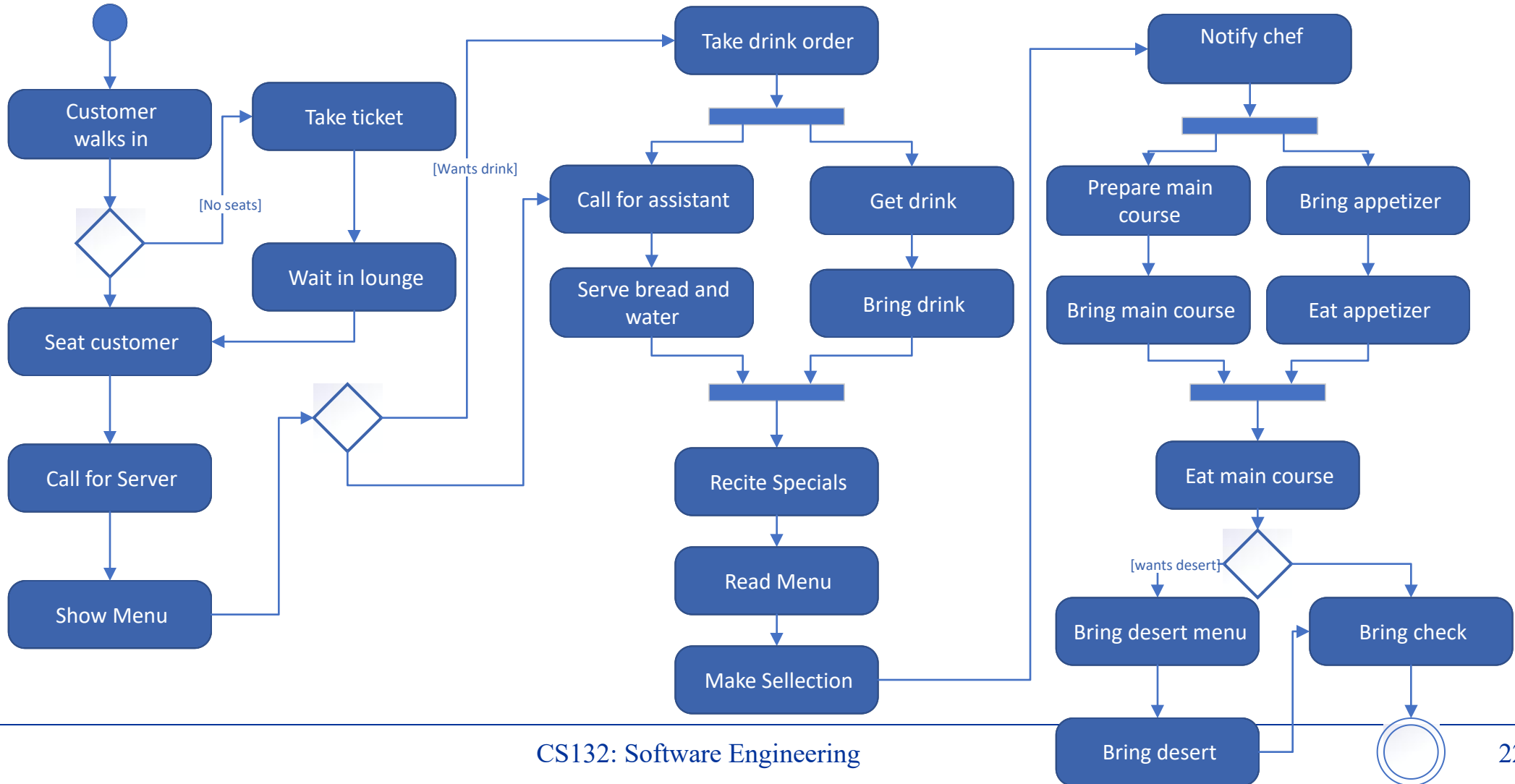


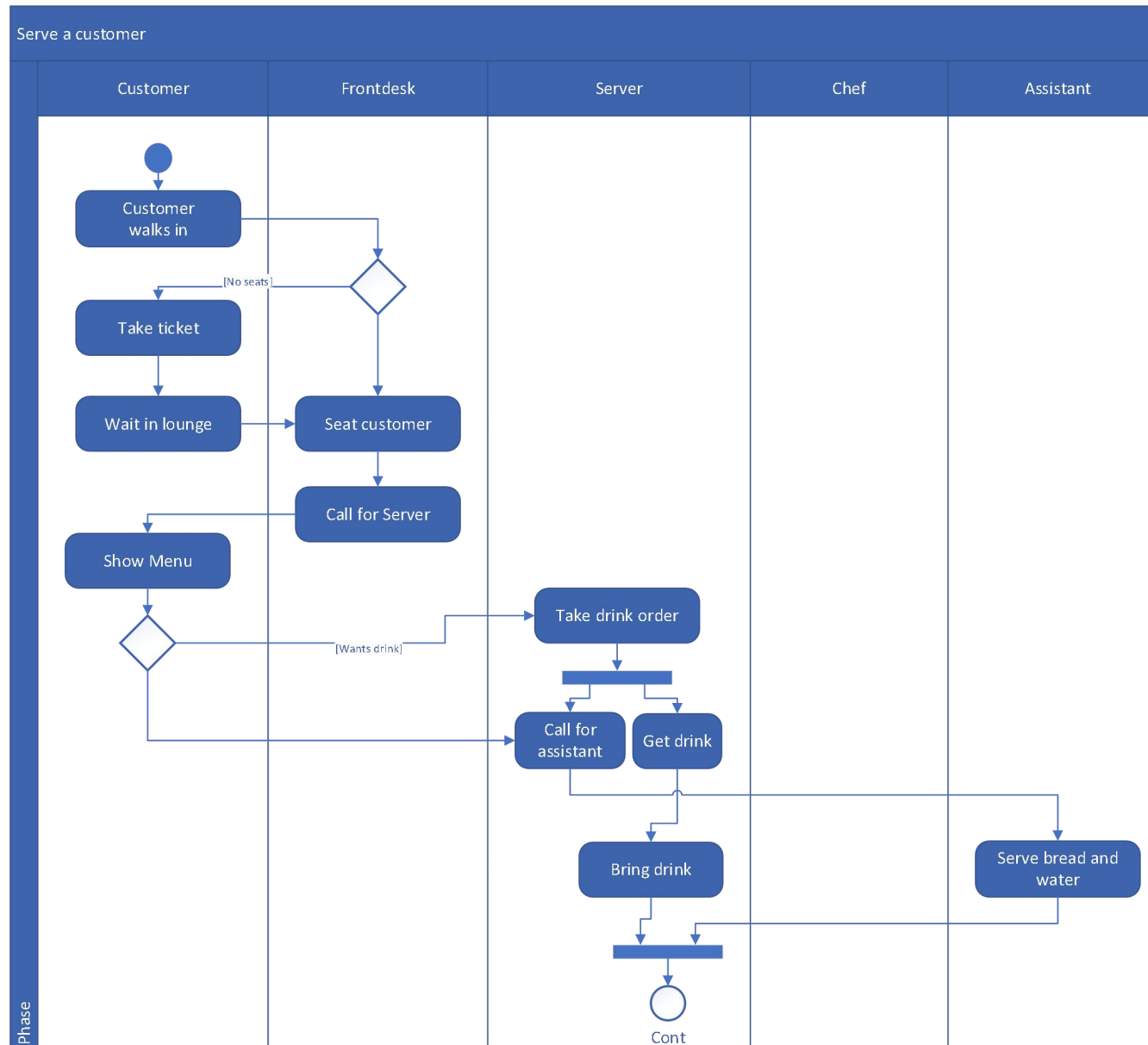
Server

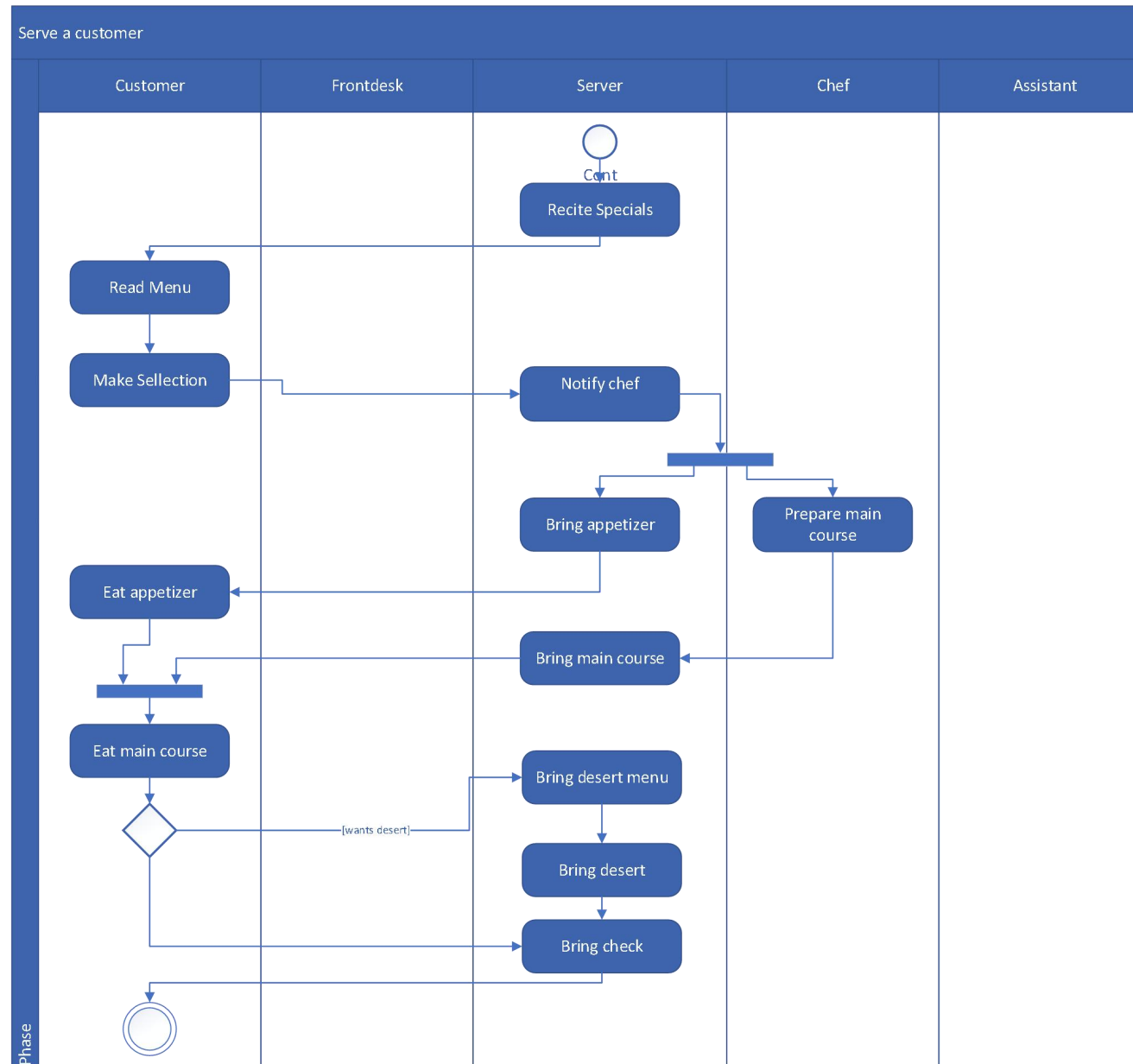


Chef

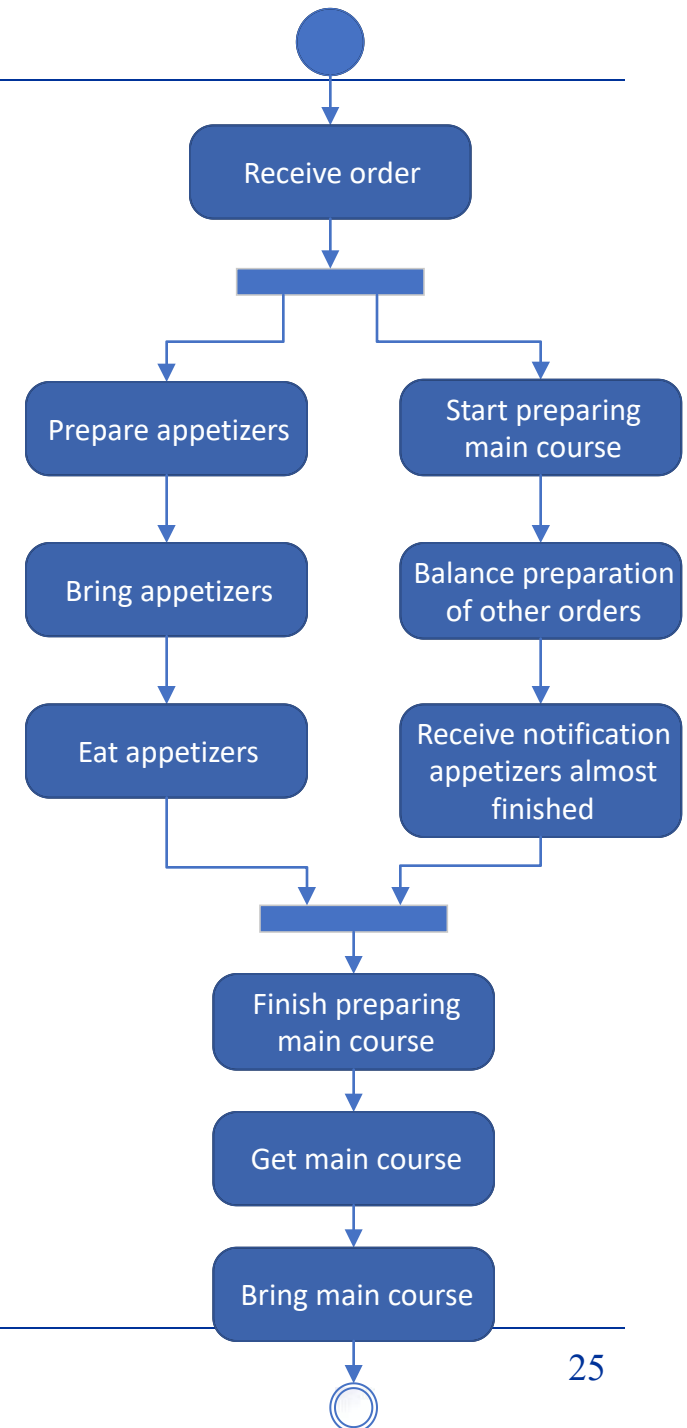
Discover Domain Procedures



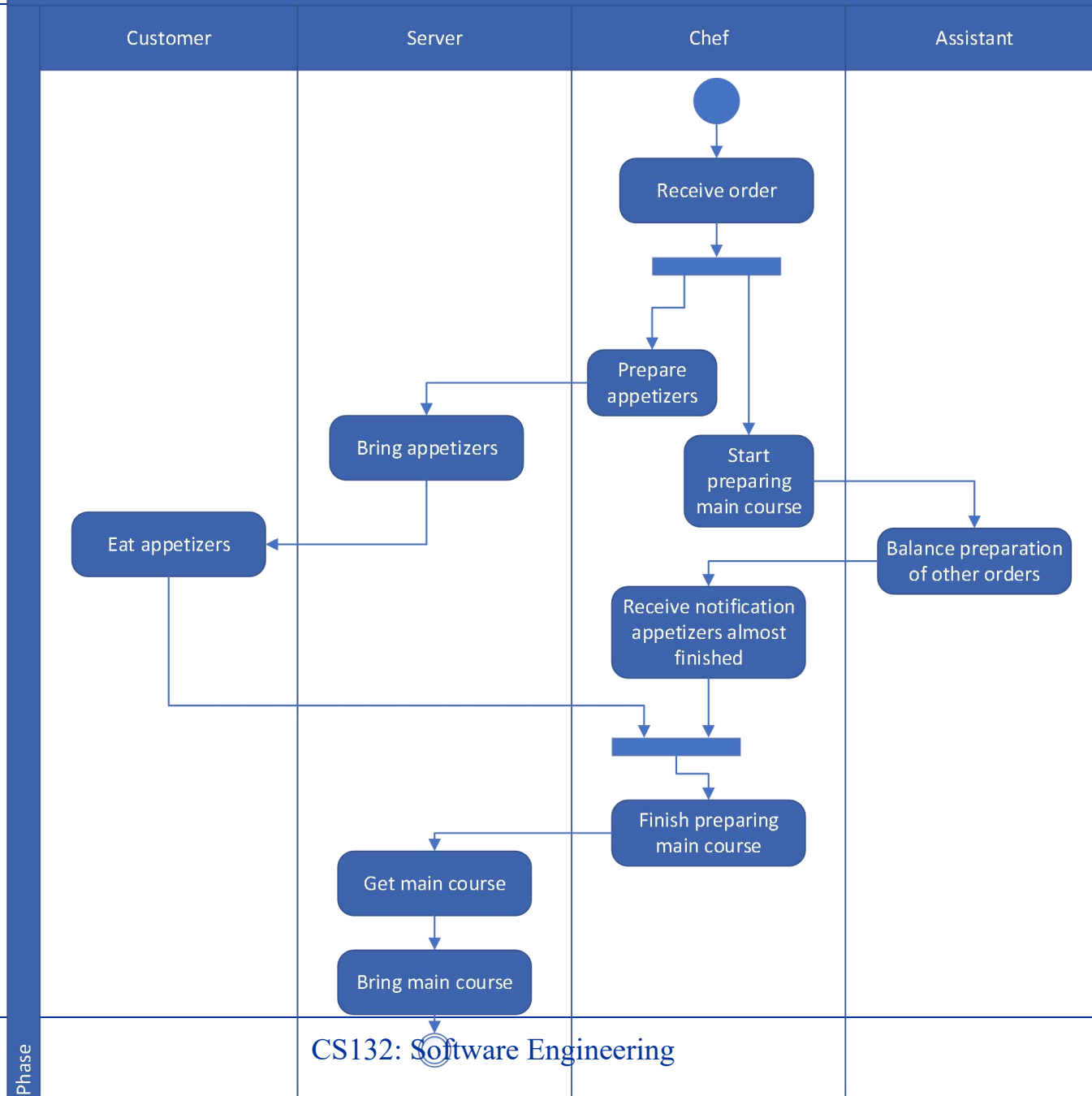




- Prepare food

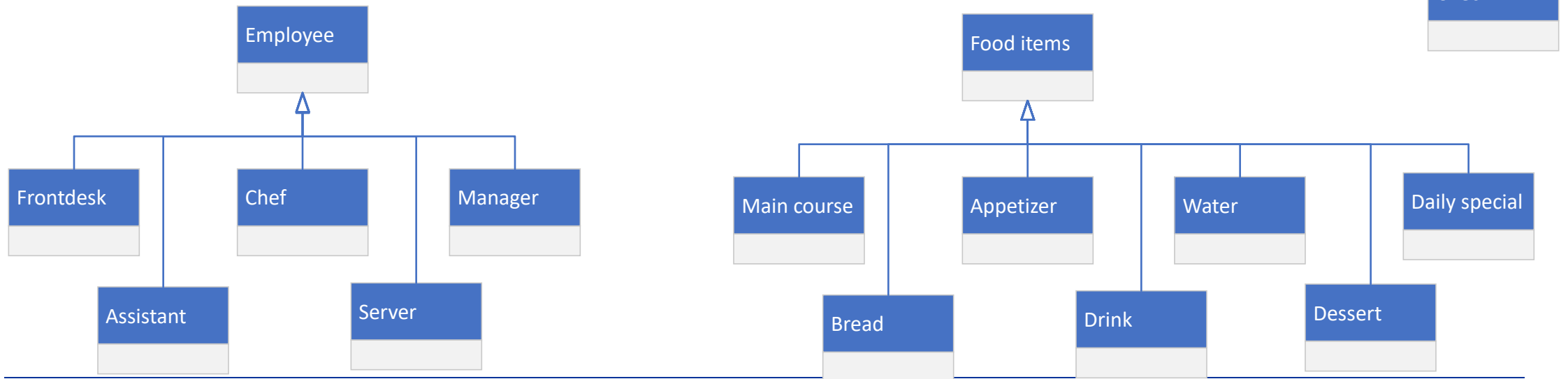


Prepare meal



Domain Analysis

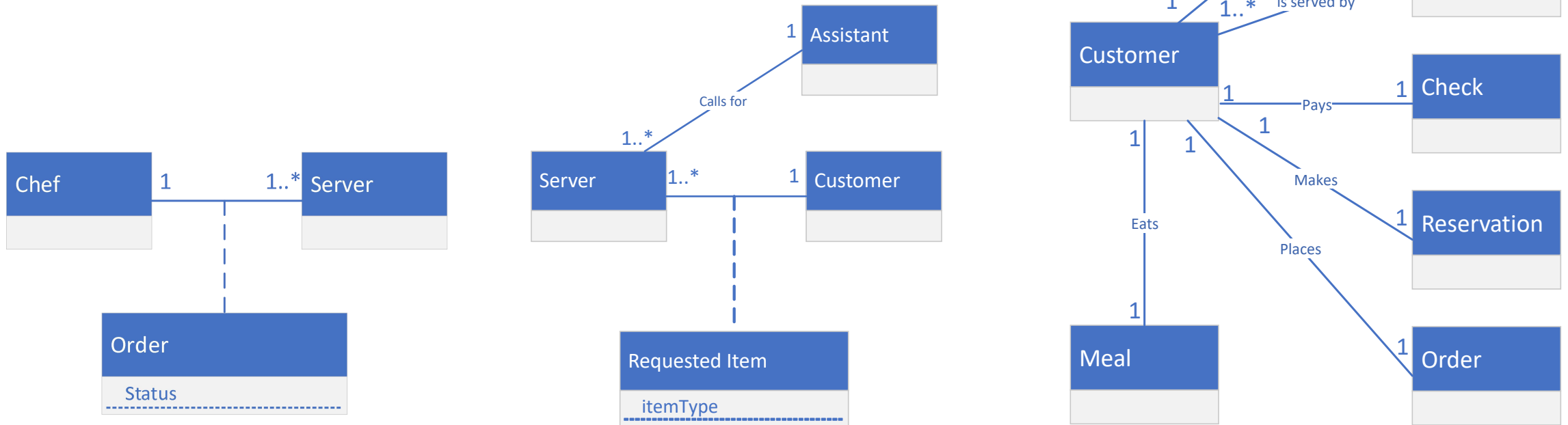
1. Develop 1st version of class diagram
2. Find similar attributes and organize objects into classes



Domain Analysis (cont.)

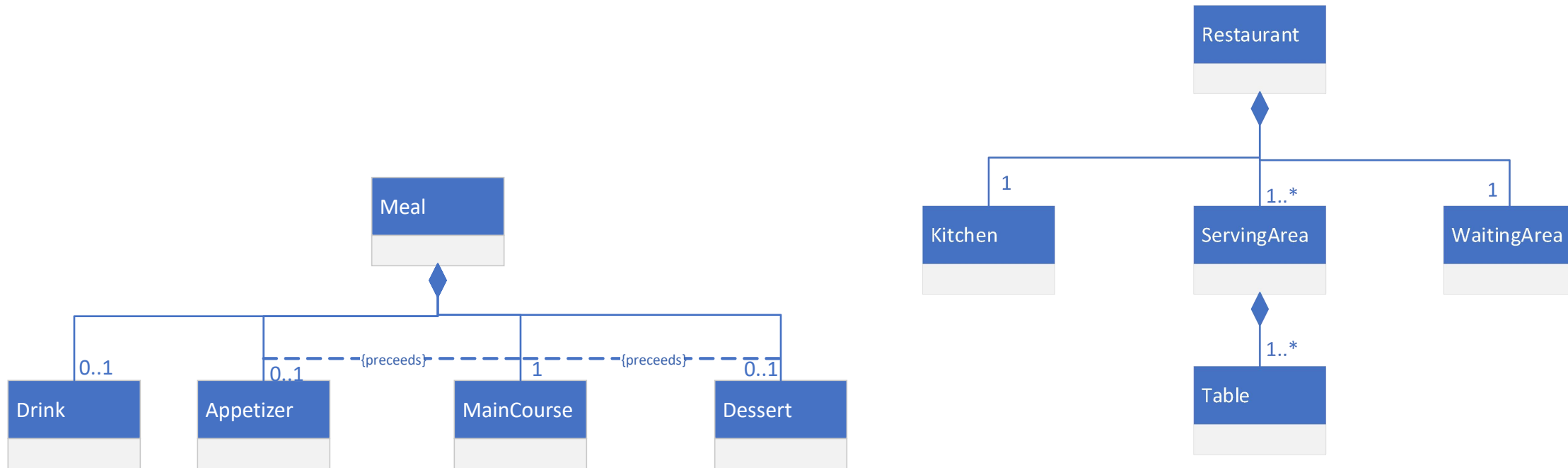
3. Further understand the domain

- Find associations



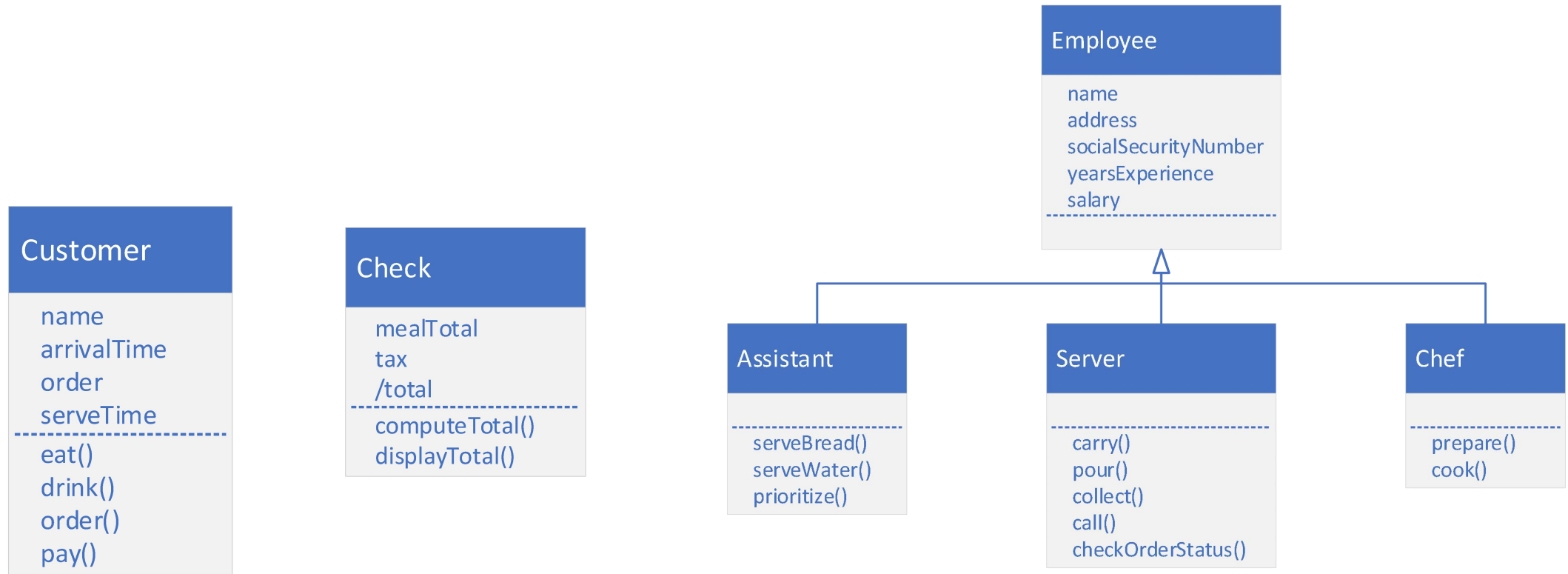
Domain Analysis (cont.)

4. Find aggregations and compositions



Domain Analysis (cont.)

5. Enrich information in classes

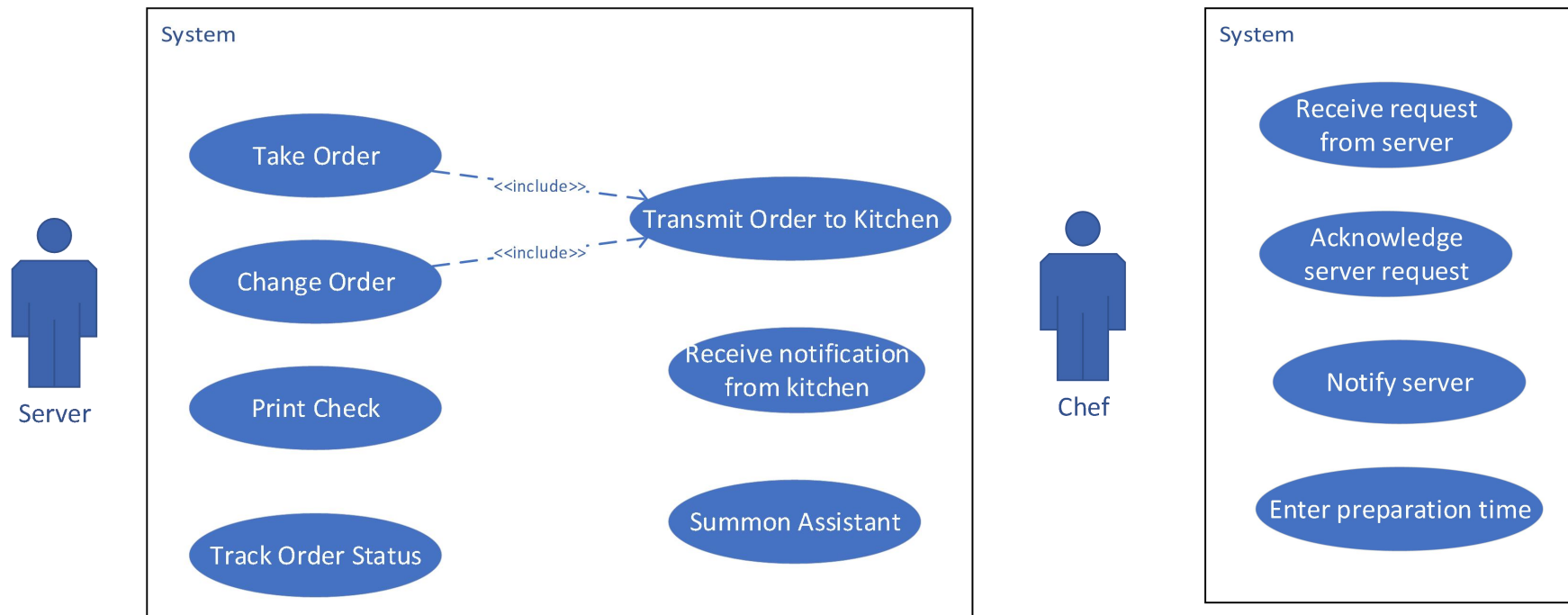


Discover system requirements

- 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

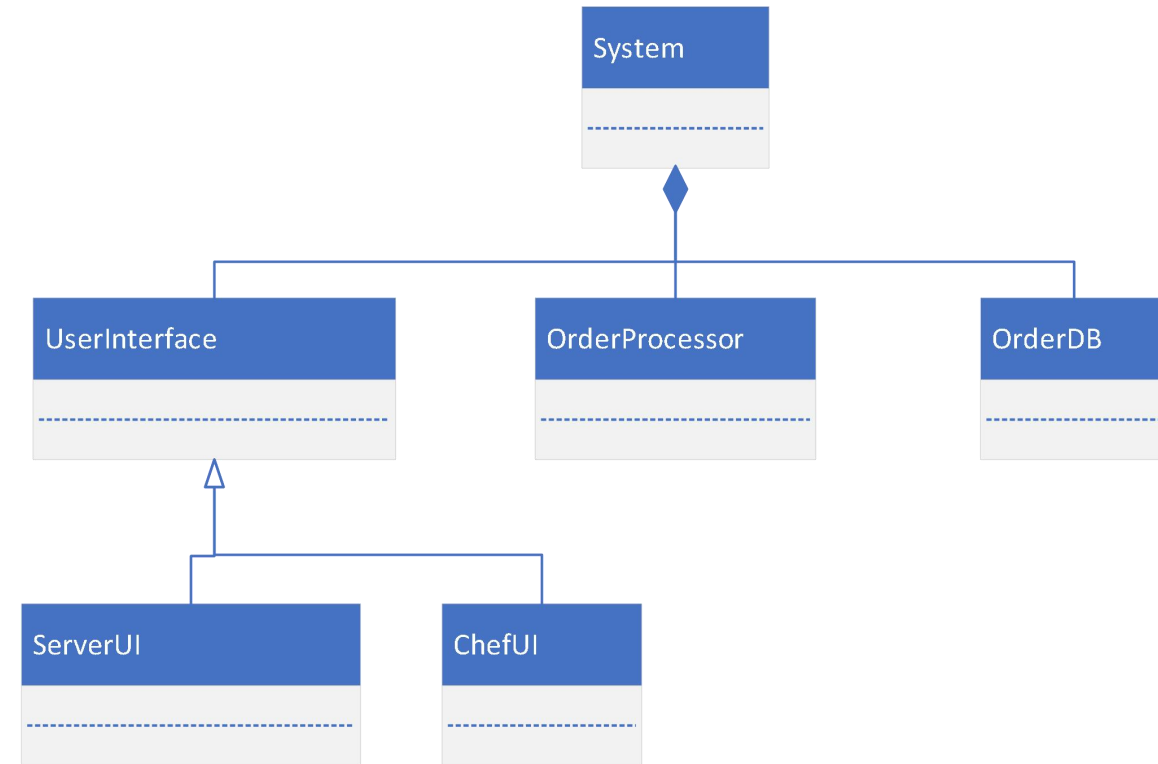
Discover system requirements (cont.)

- 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

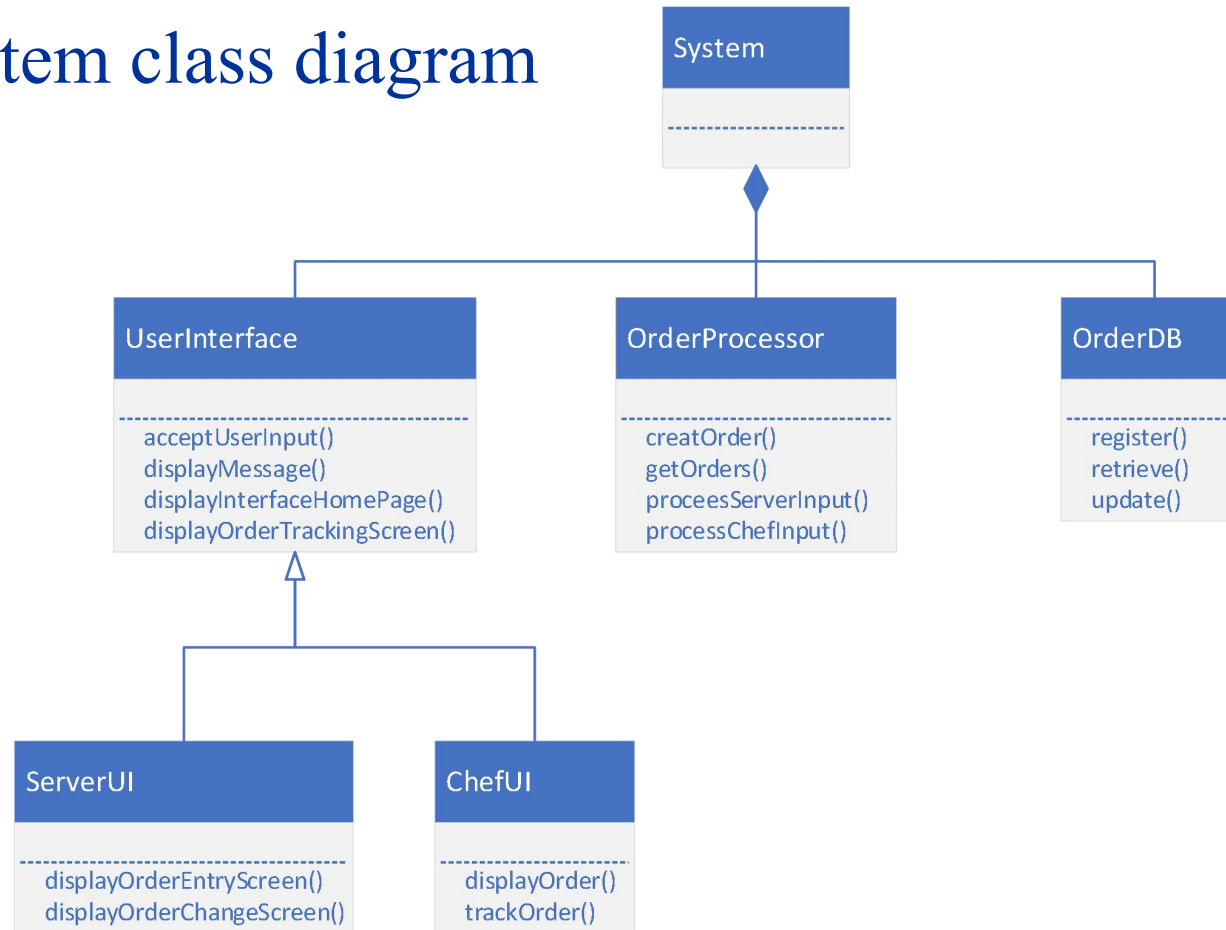


Discover system requirements (cont.)

- 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

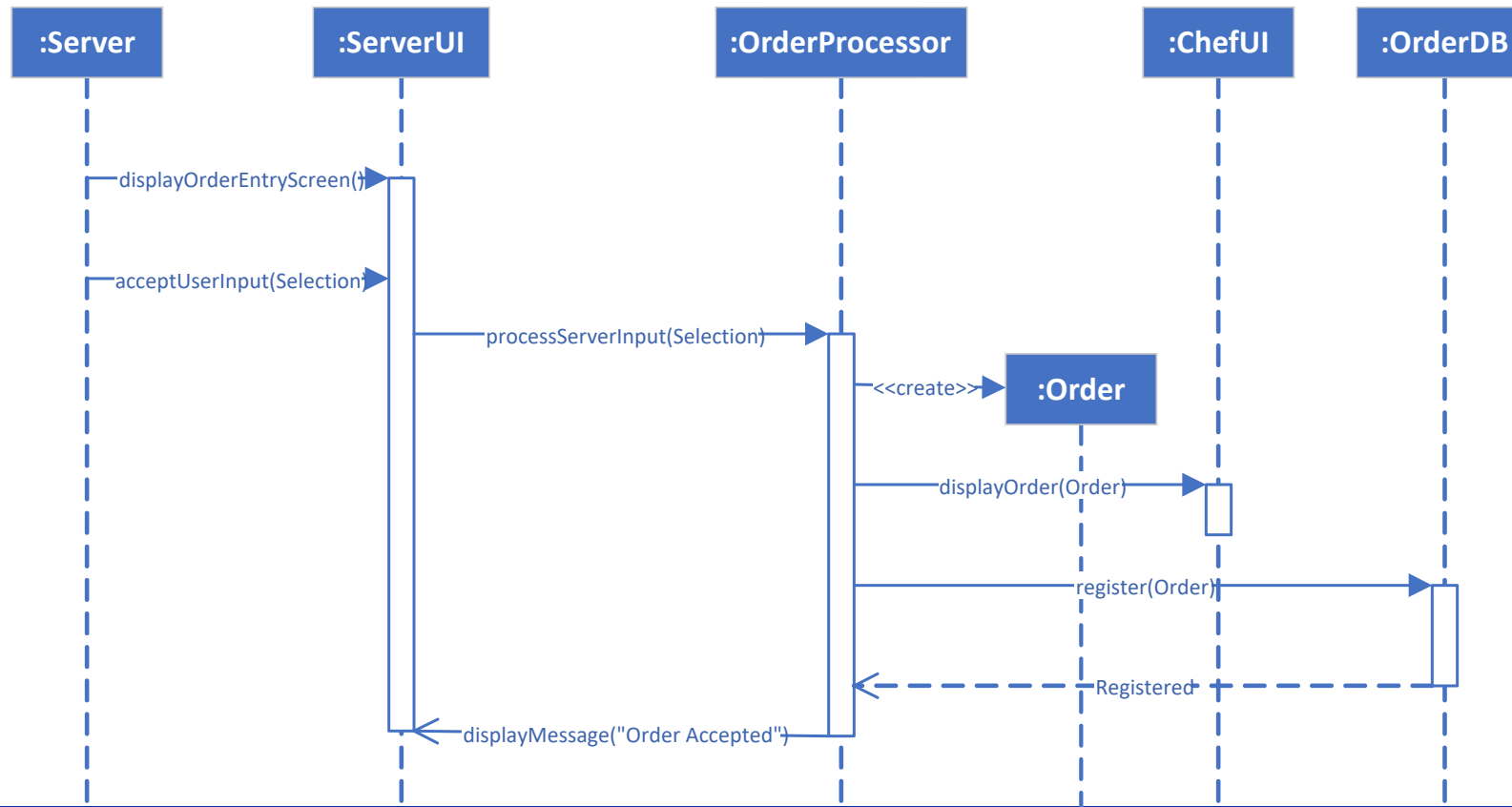
Discover system requirements (cont.)

- Enriched system class diagram



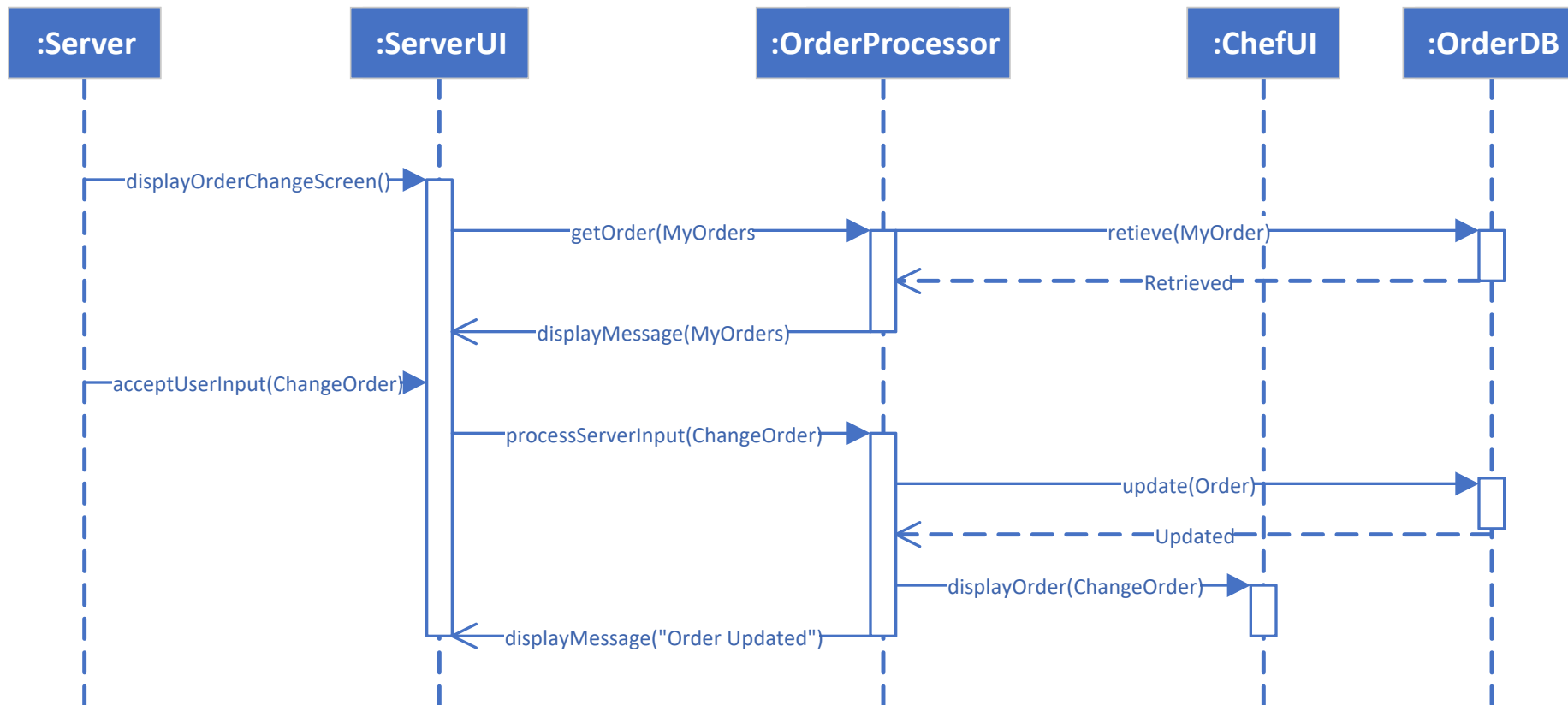
Identify interactions

- Use case “Take an order”



Identify interactions (cont.)

- Use case “Change an order”



Identify interactions (cont.)

- Use case “Track an order”

