Object Oriented Software Engineering (SE 313)
Lab Session # 03

Objective: TO UNDERSTAND SEQUENCE DIAGRAM

Objectives

- System Sequence Diagram versus Sequence Diagram.
- Notations of System Sequence Diagram.
- System Sequence Diagram and use cases.
- Exercise and Student Tasks.

System Sequence Diagram (SSD) versus Sequence Diagram (SD)

- A System Sequence Diagram is an artifact that illustrates input and output events related to the system under discussion.
- System Sequence Diagrams are typically associated with use-case realization in the logical view of system development.
- Sequence Diagrams (Not *System* Sequence Diagrams) display object interactions arranged in time sequence.

Sequence Diagram

- Sequence Diagrams depict the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the system.
- Sequence diagrams can be used to drive out testable user interface requirements.

System Sequence Diagram Behavior

- System behaves as "Black Box".
- Interior objects are not shown, as they would be on a Sequence

System Sequence Diagram

Use cases describe

- How actors interact with system
- Typical course of events that external actors generate and
- The order of the events.

For a particular scenario of use-case an SSD shows

- The external factors that interact directly with the system.
- The System (as a black box).
- The system events that the actors generate.
- The operations of the system in response to the events generated.
- System Sequence Diagrams depict the temporal order of the events.

- System Sequence Diagrams should be done for the main success scenario of the use-case, and frequent and alternative scenarios.

Notation

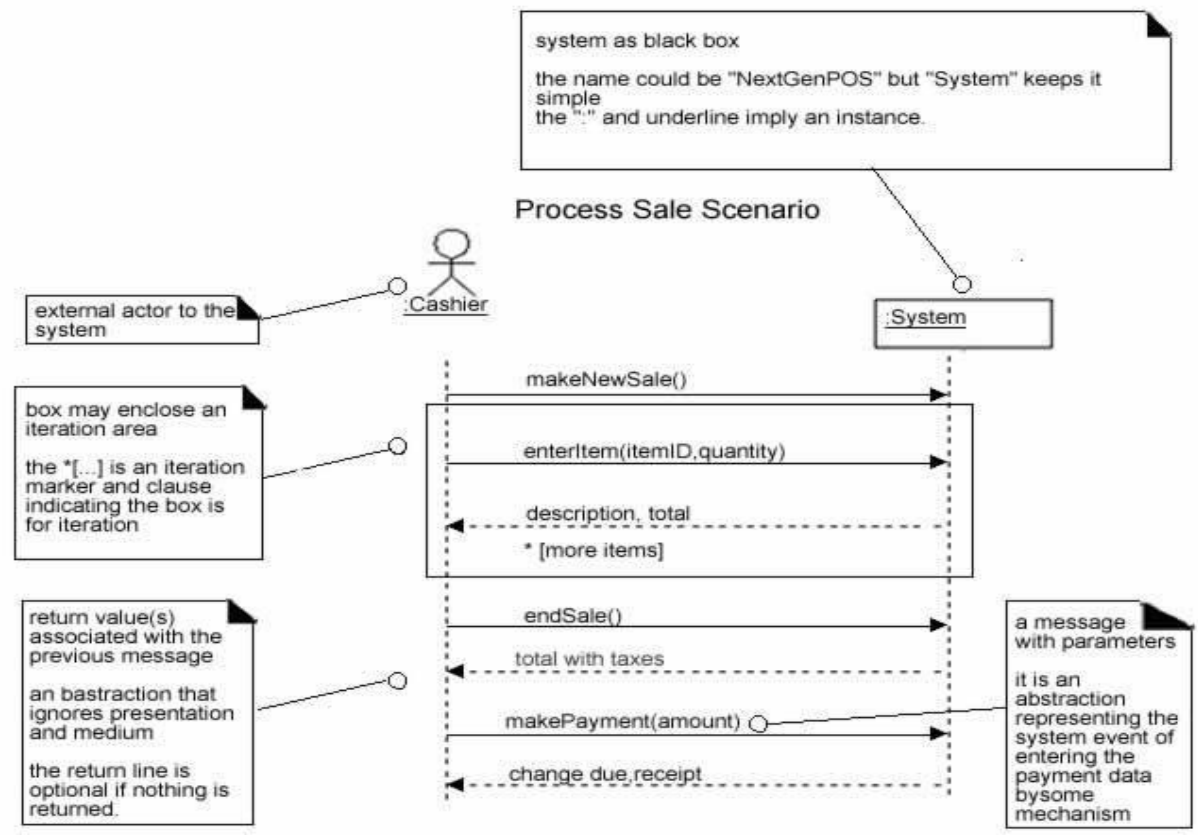
- **Object**
  Objects are instances of classes. Object is represented as a rectangle which contains the name of the object underlined. Because the system is instantiated, it is shown as an object.

- **Actor**
  An Actor is modeled using the ubiquitous symbol, the stick figure.

Example of an SSD

Following example shows the success scenario of the Process Sale use case.
Events generated by cashier (actor)

- makeNewSale
- enterItem
- endSale and
- makePayment



SSD for Process Sale scenario

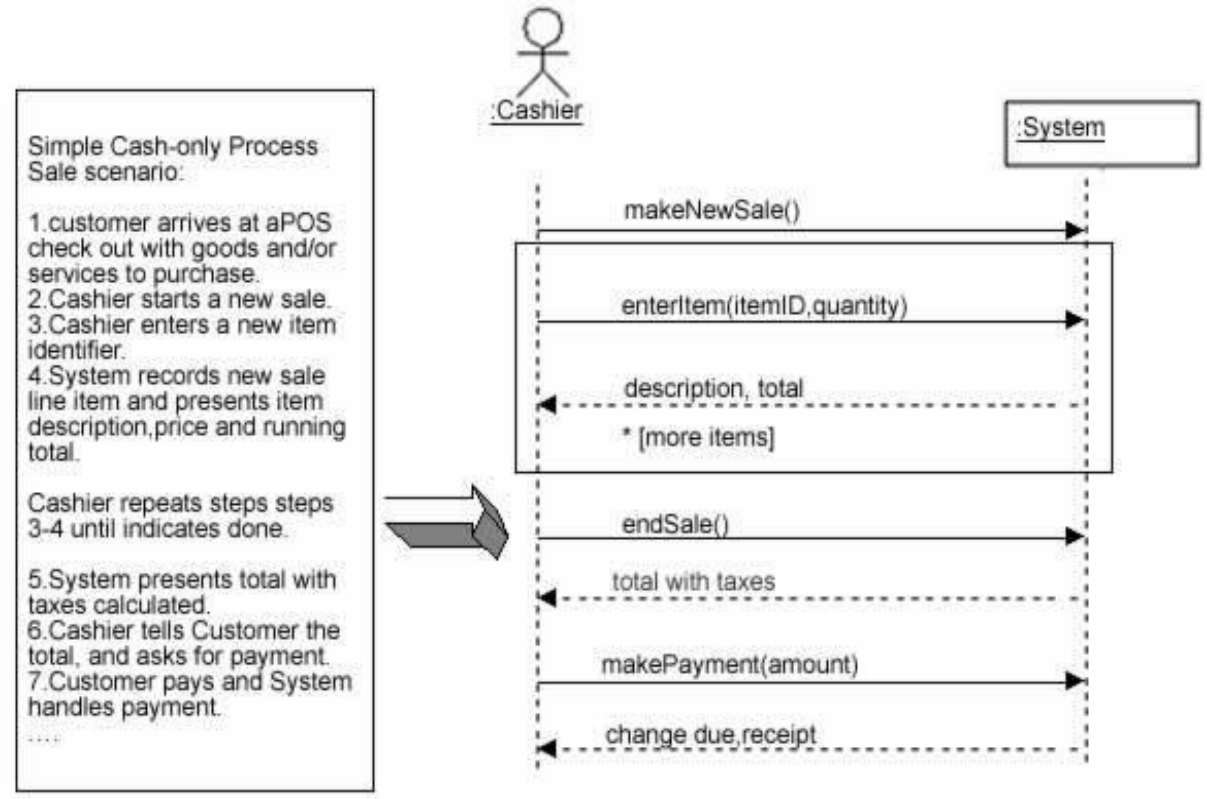
System Sequence Diagrams and Use Cases

System Sequence Diagram is generated from inspection of a use case.

Constructing a systems sequence diagram from a use case

1. Draw a line representing the system as a black box.
2. Identify each actor that directly operates on the system. Draw a line for each such actor.
3. From the use case, typical course of events text, identify the system (external) events that each actor generates. They will correspond to an entry in the right hand side of the typical use case. Illustrate them on the diagram.
4. Optionally, include the use case text to the left of the diagram.

SSDs are derived from use cases

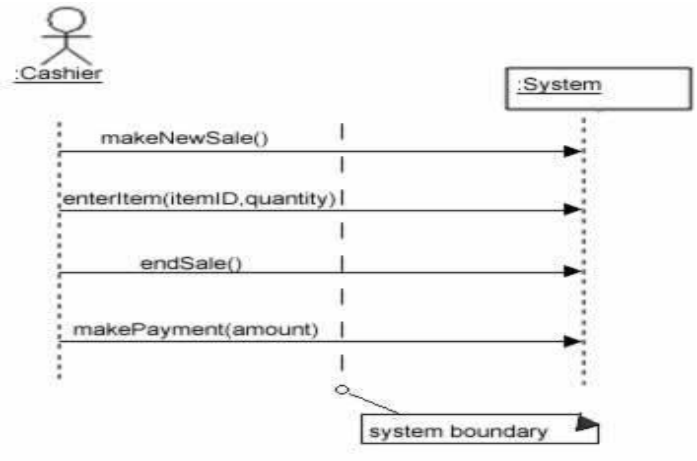


System Events and System Boundary

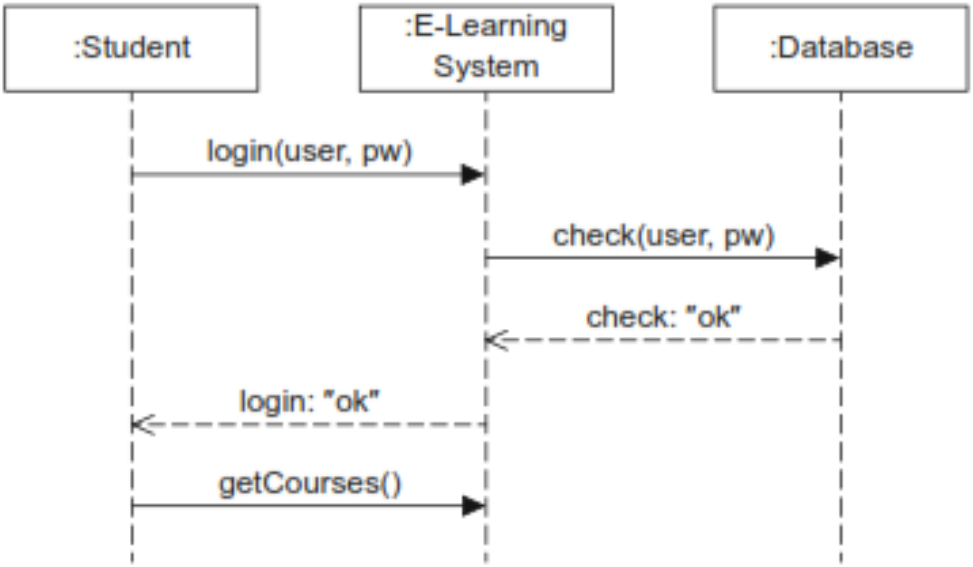
To identify the system events, system boundary is critical.
For the purpose of software development, the system boundary is chosen to be the software system itself.

Identifying the System events

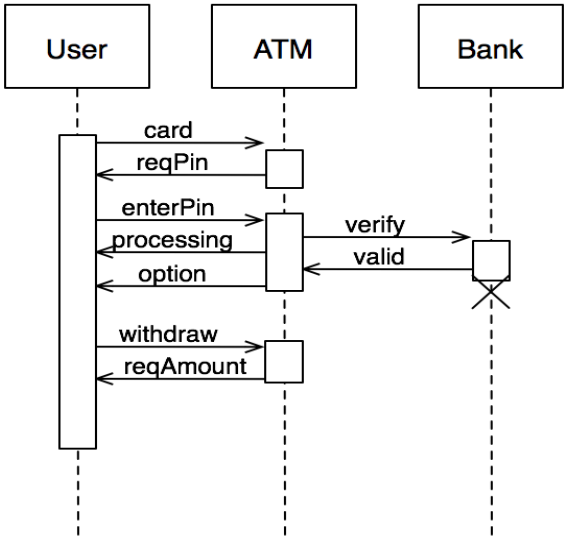
1. Determine the actors that directly interact with the system.
2. In the process Sale example, the customer does not directly interact with the POS system. Cashier interacts with the system directly. Therefore cashier is the generator of the system events.

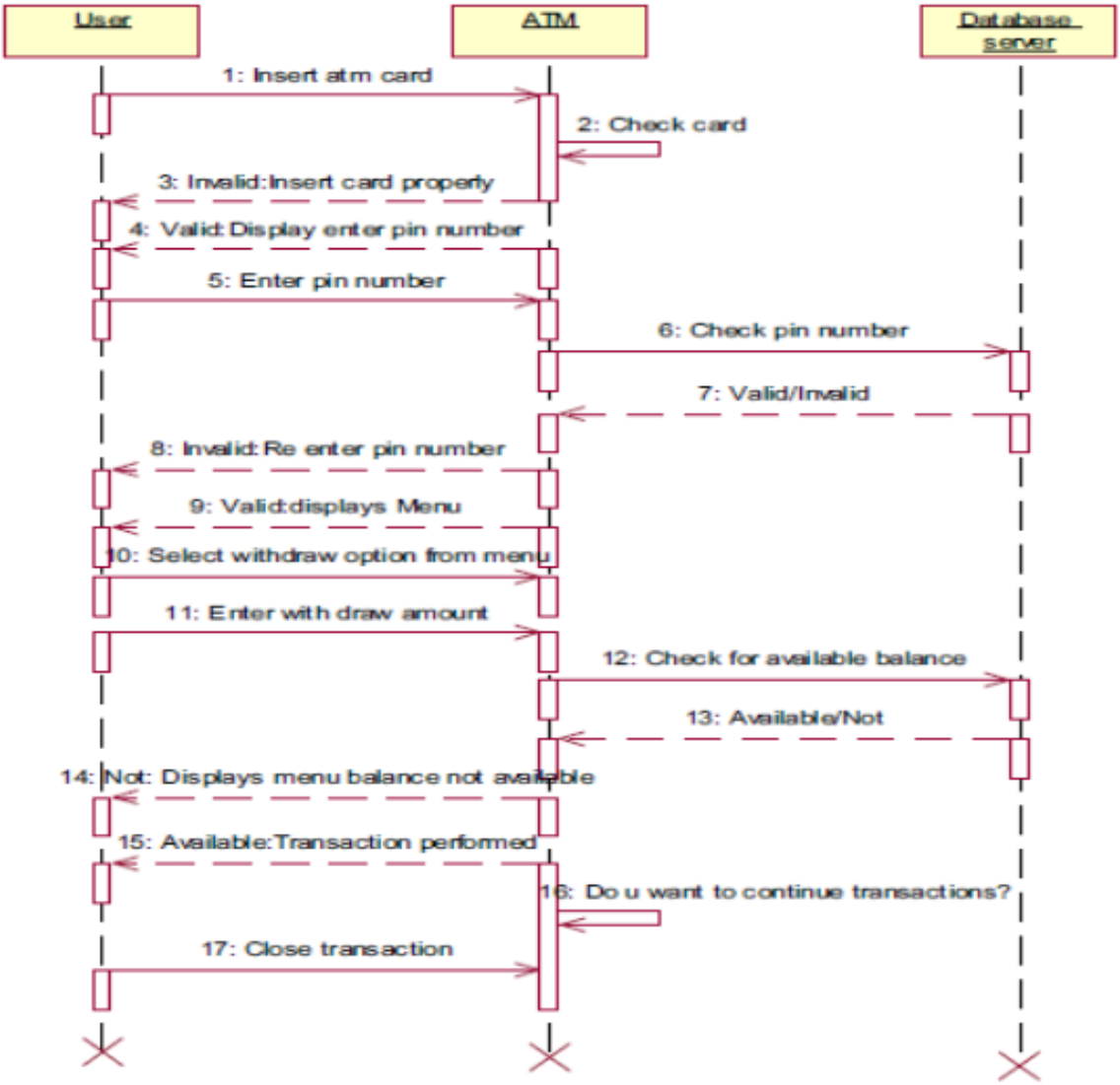


Example: Login sequence of E-Learning



Exercise 1: ATM Machine





Exercise 2: Course Registration Form

Create a new sequence diagram called Add a Course Offering for the Select Courses to Teach

- Add the Professor actor to the diagram.
- Create the following objects: course options form, add course form, course, course offering
- Create the messages as shown in the table below.

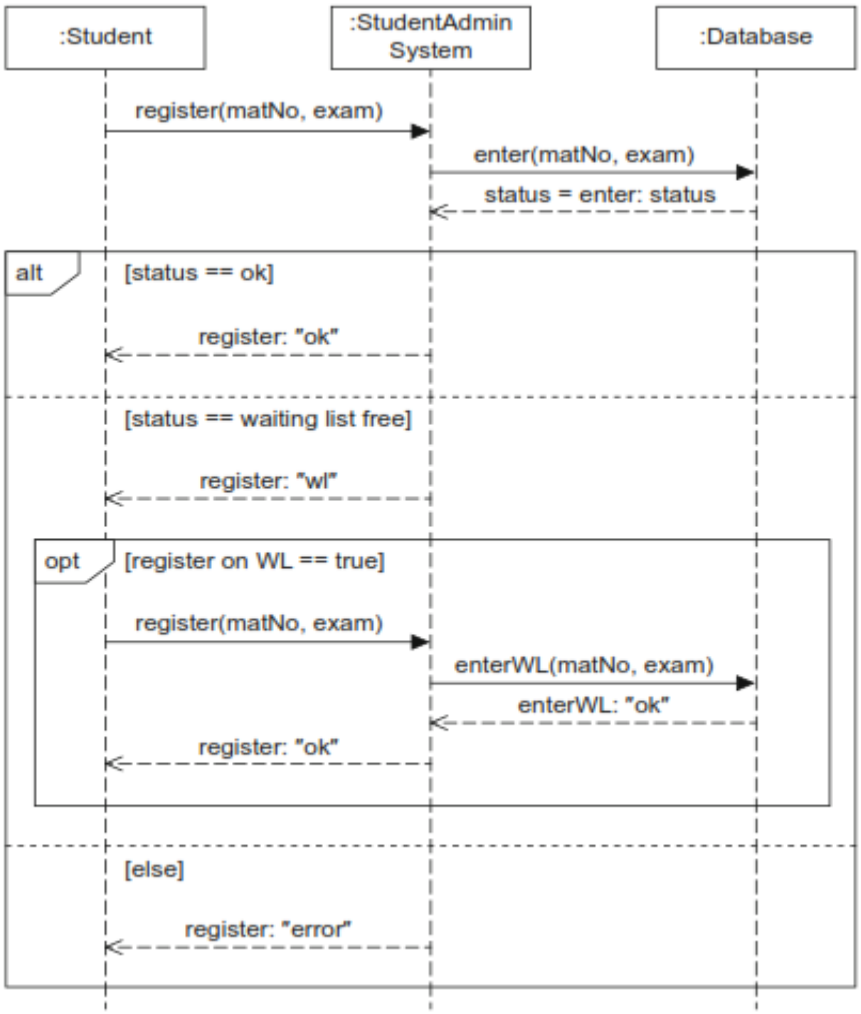
From	**To**	**Message**
Professor	course options form	add a course
course options form	add course form	display
Professor	add course form	select course offering
add course form	course	add professor (prof id)
course	course	get professor (prof id)
course	course offering	add professor (professor)

Save the model.

Branching and Looping in Sequence Diagram

Branches and loops	alt	Alternative interaction
	opt	Optional interaction
	loop	Iterative interaction
	break	Exception interaction

Example of alt and opt in sequence diagram



When a student wants to register for an exam, the following cases can occur:

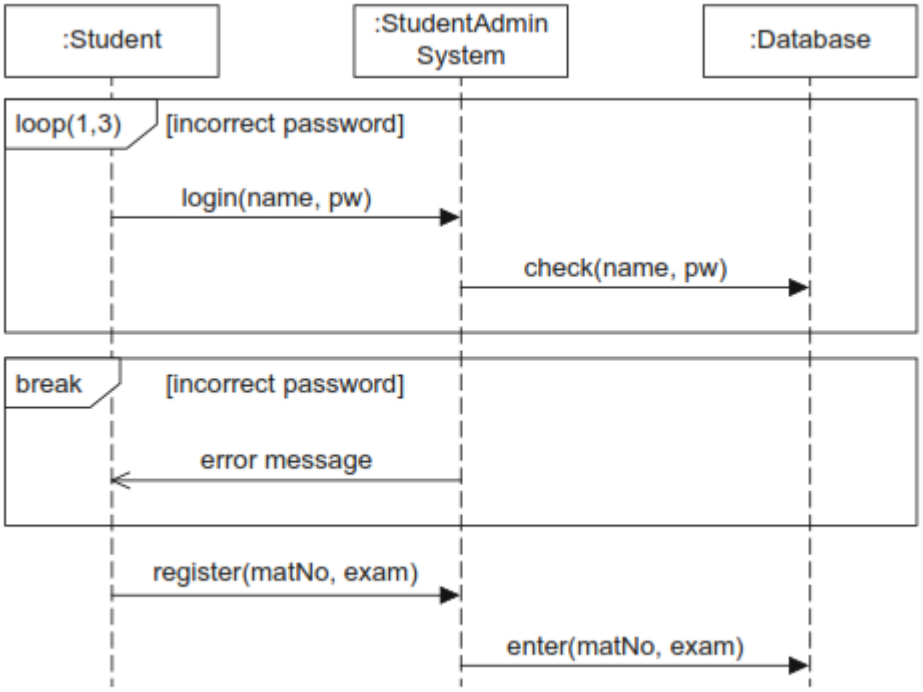
(1) There are still places available and the student can register.

(2) There is a place available on the waiting list. Then the student has to decide whether to go on the waiting list.

(3) If there is no place available for the exam or on the waiting list for the exam, the student receives an error message and is not registered for the course.

This would be specified as an if statement without an else branch. Figure illustrates the use of the **opt** fragment. If there is a place available on the waiting list, when registering for an assignment the student can decide whether to take the place on the waiting list. If the student wants to be on the waiting list, the student has to register for it.

Example of Loop and Break in sequence diagram



System login that is necessary before a student can register for an assignment. The password must be entered at least once and at most three times, as reflected by the arguments of **loop**. After the first attempt, the system checks whether the password can be validated. If it can, that is, the condition Password incorrect is no longer true, execution of the interactions within the **loop** ceases. The system also exits the **loop** if the student enters the password incorrectly three times. This case is then handled further in the subsequent break fragment.

The **break** operator thus offers a simple form of exception handling. For our example in Figure above, this means that if the password is entered incorrectly three times, the condition incorrect password is true. Thus the content of the **break** fragment is executed, meaning that an error message is sent to the student and the student is not allowed to register for the assignment. The remainder of the interaction after the end of the break fragment is skipped. After exiting the break operator, we are in the outermost fragment of the sequence diagram and therefore the execution of this sequence diagram is ended. If we were not in the outermost fragment, the sequence diagram would continue in the fragment at the next higher level.

Student Tasks

Create Sequence diagram for the following system

Task 1 : Hospital Management system

a) Use Receptionist as an actor with System and Database as objects
b) Draw design level sequence diagram (use all objects reception actor)

Task 2 : Airline Reservation System

Draw design level sequence diagram covering use cases (reserve a flight, book a flight, search a flight, show flight status, check rewards points & cancel flight)