

Analysis (part III)

- Dynamic Models are abstractions of:
 - What happens.
 - When it happens.
- Dynamic models in analysis include:
 - Sequence Diagrams.
 - State Transition Diagrams.
 - Activity Diagrams.

1

Analysis vs. Design

- What we should be describing at analysis time:
 - What the problem space objects are.
 - What their relationships are.
 - Their attributes and operations.
 - Their states and state changes.
 - How the objects collaborate to accomplish the system operations.

2

System Operations (1 of 2)

- Behaviors that are visible from outside the system.
- Describe what the system does.
- Elementary operations (from the outside perspective)
 - Can't be decomposed.
- Invoked by the actors during scenarios.
- Convenient level of granularity for interaction diagrams.

3

System Operations (2 of 2)

- The only way to change the system state is to invoke a system operation.
- The invocation of a system operation is called a "trigger."
- We can find the system operations in the scenario descriptions.
 - Message from an actor to the system.
 - System performs the system operation.
 - System may send messages to other actors.
 - System returns to the invoking actor.

4

System State Changes

- All information must be stored as either:
 - Attributes of object instances (or links).
 - Links between object instances.
- Therefore the only kinds of system state changes are the following:
 - Creation of an instance of a class.
 - Destruction of an object instance.
 - Creation of a link between two object instances.
 - Destruction of a link between two objects.
 - Modification of the values of the attributes of an object instance (or link).

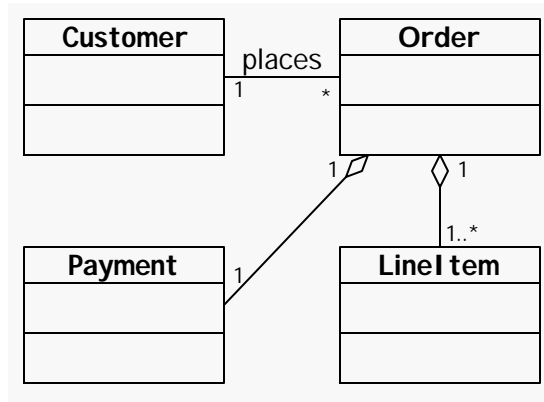
5

Example (1 of 6)

- Consider an Internet storefront application.
- Some system operations might be:
 - Create a new order.
 - Add a line item to an order.
 - Delete an item from an order.
 - Check out.

6

Example (2 of 6)



7

Example (3 of 6)

- System Operation: Create New Order
 - Name: Create New Order.
 - Description: Start and new order for a customer.
 - Actors: Customer.
 - Inputs: Customer name and address.
 - Objects Involved: Customer, Order.
 - Preconditions: System up and running.

8

Example (4 of 6)

- Post-conditions: If existing customer, new order is created and linked to the customer object, else a new customer object is created and a new order is created and is linked to that customer object.
- Outputs: Welcome message.

9

Example (5 of 6)

- System Operation: Add Order Item.
 - Name: Add Order Item.
 - Description: Add a line item to an order.
 - Actors: Customer (p), Inventory (s).
 - Inputs: Item number, quantity.
 - Objects Involved: Order, LineItem.
 - Preconditions: Order exists.

10

Example (6 of 6)

- Post-conditions: If sufficient inventory exists, then a LineItem is created and is linked to the Order.
- Outputs: If sufficient inventory exists, then reply OK to the customer, else reply with an error message indicating the problem.

11

Use Cases and System Operations

- Use Cases are dynamic models that treat the system as a black box.
- They describe the system from the *outside*, because everything inside the system is hidden (at Requirements time).
- System Operations specify the system from the *inside* because they describe the state changes in terms of objects within the system boundary.

12

Describing System Operations

- Three dynamic analysis models can be used:
 - Sequence Diagrams.
 - Communication Diagrams.
 - Activity Diagrams.
- All three models show:
 - What objects are involved in a system operation.
 - What their responsibilities are.
 - What the flow of control (sequence of messages) is within the system.

13

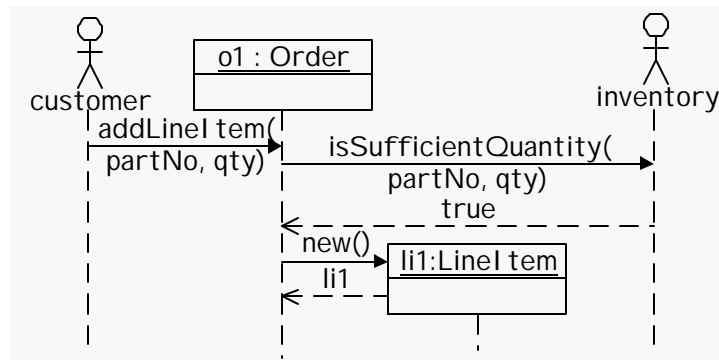
Sequence Diagrams (1 of 2)

- Show the object *instances* that are involved in the system operation, and exist at the beginning of the system operation, along the top of the page.
- Draw a dashed line (lifeline) below each object instance box.
- Indicate messages as arrows from one dashed line to another.
 - Label the arrow with the name of the operation (and possibly arguments passed).
- Indicate returns as dashed arrows.
- Time goes down the page.

14

Sequence Diagrams (2 of 2)

- Add Order Item (itemNo, quantity)



15

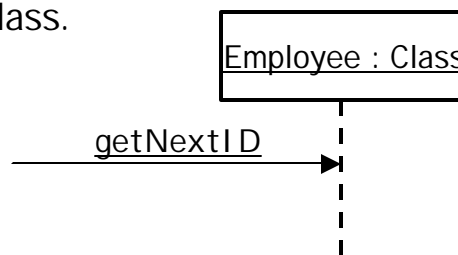
Object Instance Creation and Destruction

- Creation: arrow to the object instance box. The object comes into existence at this point in the timeline.
- Destruction: Put a large X on the lifeline, and discontinue the lifeline below the X.

16

Invoking a Classifier Operation

- Since the lifelines all represent object instances, how do you show a message being sent to a class?
- A class is actually an object instance of the metaclass, Class.



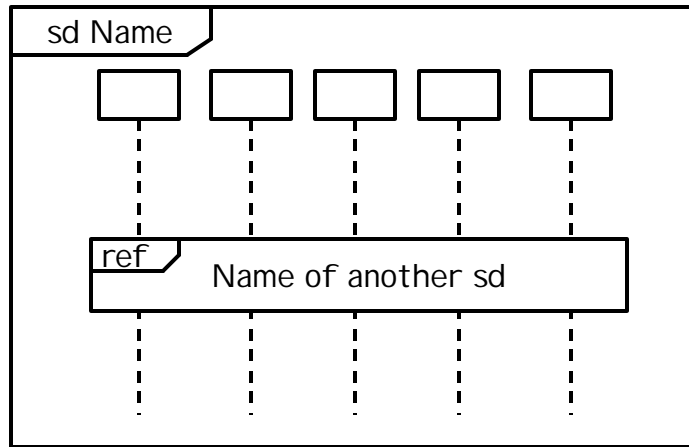
17

Managing Sequence Diagrams with Frames

- A sequence diagram may be enclosed in a frame.
 - Rectangle, with a pentagon in the upper left corner (the heading).
 - The contents of the heading pentagon indicates what kind of frame it is.
- Frames allow for:
 - Modularity.
 - Looping.
 - Decisions.
 - Parallelism.
 - etc.

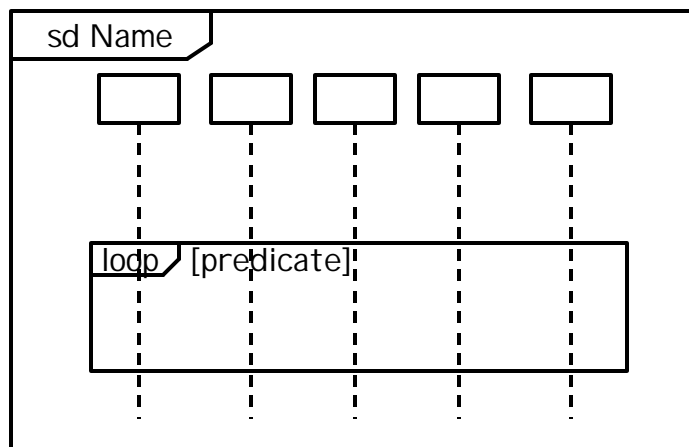
18

Modularity



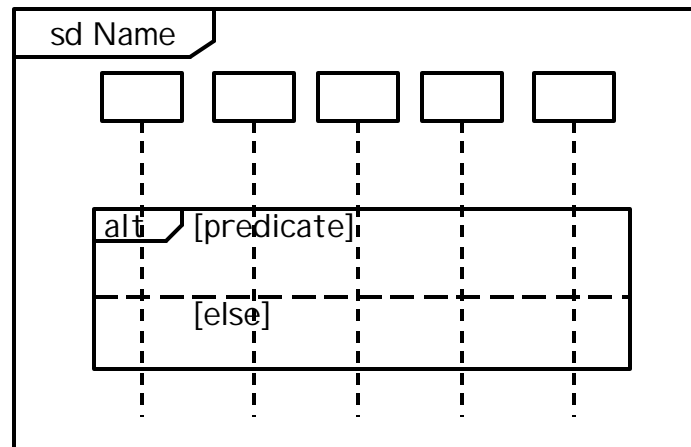
19

Looping



20

Decisions



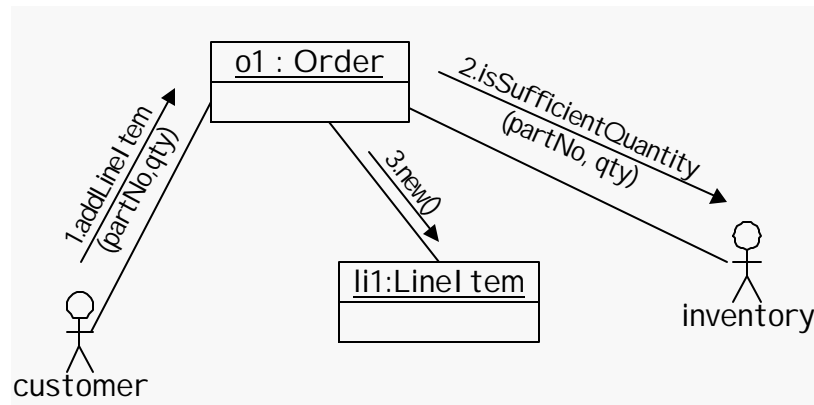
21

Communication Diagrams (1 of 2)

- Show the objects involved in the scenario spread out on the drawing surface.
- Indicate messages from one object to another as arrows, labeled with the operation and arguments.
- Indicate returns as dashed arrows.
- Indicate time sequence by numbering the arrows.

22

Communication Diagrams (2 of 2)



23

Activity Diagrams (1 of 2)

- Bubbles indicate activities (for our purposes, operations executed by objects).
- Arrows indicate flow of control.
 - May be labeled with a conditional guard.
- Start and stop indicated by bullets.

24

Activity Diagrams (2 of 2)

- Forks and joins for parallelism.
- Decision boxes for conditional execution and looping.
- In *any* path through the activity diagram, the post-conditions of one activity must lead to the preconditions of the following activity.
- Partitions show which object or subsystem is responsible for which activities.

25

State Modeling (1 of 2)

- Each object instance has its own life cycle.
 - It is created.
 - It is initialized.
 - It receives some sequence of messages.
 - Its attributes go through some corresponding sequence of states.
 - It finally may go out of existence.
- All object instances of the same class obey the same rules.
- These rules are documented in a State Machine Diagram (SMD).

26

State Modeling (2 of 2)

- All subclasses of a class inherit its state related behavior.
- Subclasses may have additional states and transitions (because they may have additional attributes and operations).

27

SMD Terminology (1 of 2)

- State – An abstraction of the values of the attributes of an object, and its links to other objects.
 - The abstraction has some semantic meaning.
 - For example, the states of a flight might be atGate, taxiing, takingOff, climbing, levelFlight, descending, landing.
 - The particular values of attributes (e.g., speed) and links (e.g., to Gate) are abstracted.

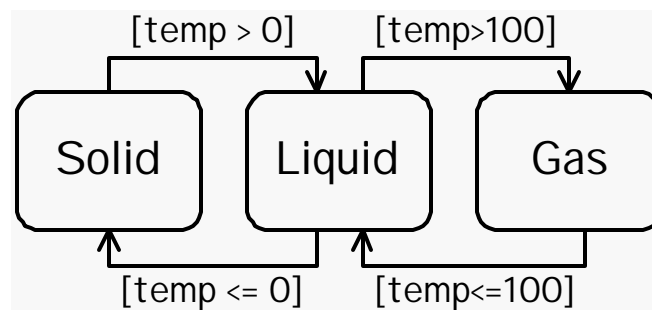
28

SMD Terminology (2 of 2)

- Event - Something that takes place at a particular instant in time. Events are considered to be instantaneous.
- Transition - What the object does in response to an event.
- State Machine Diagram - A model showing all of the states of an object, the legal transitions, and events that trigger these transitions.

29

SMD Example: H₂O at Standard Pressure



30

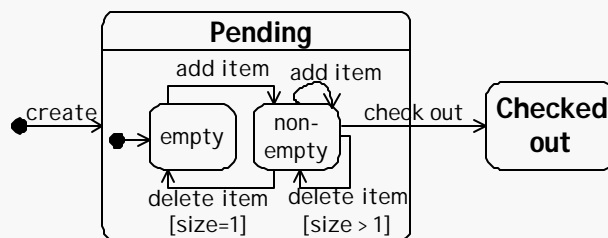
State Machine Diagrams (1 of 7)

- Notation invented by David Harel in 1987.
 - Called Harel Statecharts.
- Adopted by both Booch and Rumbaugh.
- Permit hierarchical development (through stepwise refinement) of state diagrams.
 - States may be a various levels of abstraction.
 - Within a particular state diagram, all states are at the same level of abstraction.
 - Any of these states may be further refined as a lower level state diagram (states within states).

31

State Machine Diagrams (2 of 7)

- Example: while the order is in pending state, it may undergo lower level state changes.



32

State Machine Diagrams (3 of 7)

- Initial state – shown as black bullet (birth state).
 - There must be exactly one initial state.
- Final states – shown as bullets with halos (death state).
 - There may be any number of final states.
- Guarded transitions – transitions that fire only when both the event occurs and the guard is true.
- Action – instantaneous operation.
 - entry/
 - exit/
 - Transition action.

33

State Machine Diagrams (4 of 7)

- For most transitions, the action that takes place is considered to be instantaneous, and occurs when the transition fires.
 - The order is: exit/, transition, entry/.
- Transitions fire because of either:
 - A message is received by the object.
 - A return comes back from a message that was sent.

34

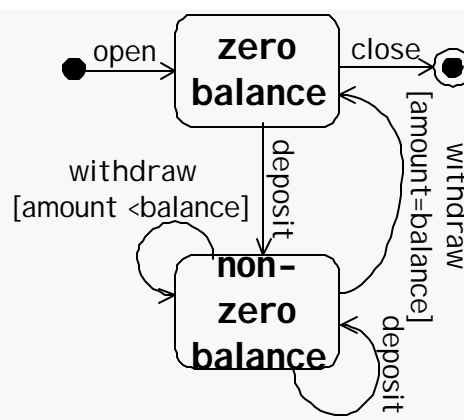
State Machine Diagrams (5 of 7)

- Some states are called *active states*.
When the object enters an active state, an activity is performed.
 - Indicted by "do/" followed by the activity name.
- An activity takes some amount of time to execute.
- When the activity finishes, there may be an automatic transition to another state.
 - (This transition does not have a label.)
- The activity may also be terminated by an event.

35

State Machine Diagrams (6 of 7)

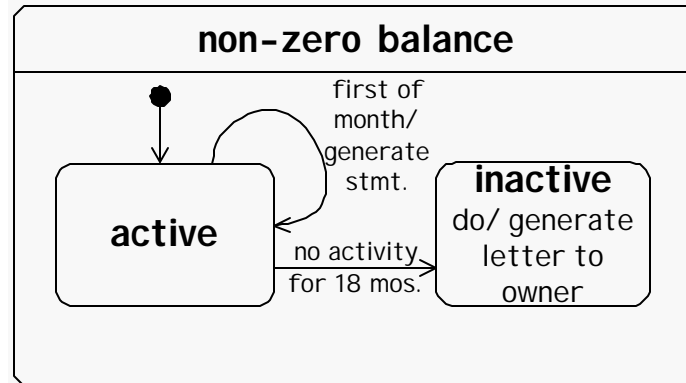
- Example: Account Class.



36

State Machine Diagrams (7 of 7)

- Detail of the Non-Zero Balance State



37

State Modeling Heuristics

- Only do state models for classes that have interesting state-related behavior.
- Use stepwise refinement for really complex classes.
- Separate objects may be thought of as operating independently. They may use messages to synchronize.
- If there is no defined transition for a particular event in some state and that event occurs, the event is ignored.

38

Other Uses of Dynamic Models

- Requirements Specification.
 - Sequence diagrams, collaboration diagrams and activity diagrams can be used instead of detailed scenario descriptions. The actors and the system play the roles of objects.
 - State transition diagrams can be used to indicate the states the system may be in, and the legal transitions between states. The transitions are triggered by the actors.
- GUIs may be modeled with STDs.

39