OO System Models

Dynamic View

UML Sequence, Communication, State Diagrams

Objective

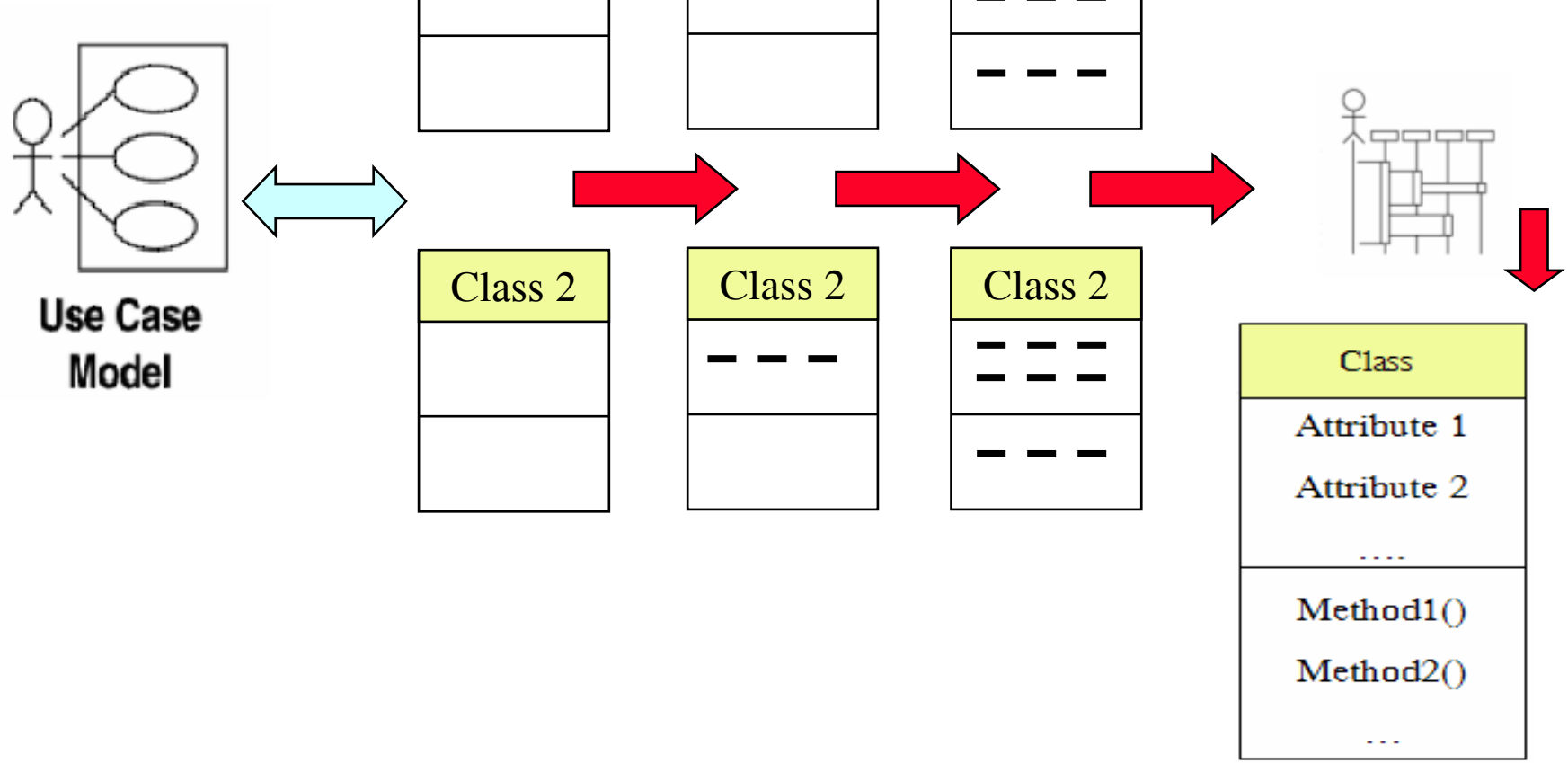
- Describe the evolutionary approach for using sequence diagrams to complete class diagrams
- Introduce the syntax of sequence diagrams
- Explain how message are interleaved between objects to build sequence diagrams
- Introduce CRUD Analysis & CRUD matrix
- Introduce communication (collaboration) Diagrams
- Explain state machine models for real time systems

OO Behaviour Modelling

The Dynamic View of a system may be described using UML diagrams:

- **UML Sequence Diagrams**
- **UML Communication 'UML 2.0' (Collaboration) Diagrams**
- **UML State Diagrams**
- **UML Activity Diagrams**

From Use Cases to: Objects, Attributes, Operations (methods) - "evolutionary"



Identifying objects

- Look for **nouns** in the SRS (System Requirements Specifications) document
- Look for **NOUNS** in use cases descriptions
- A **NOUN** may be
  - Object
  - Attribute of an object

Identifying Operations 'methods'

- Look for verbs in the SRS (System Requirements Specifications) document
- Look for **VERBS** in use cases descriptions
- A **VERB** may be
  - translated to an **operation** or set of operations
  - A method is the code implementation of an operation.

OO: Visibility of attributes or operations

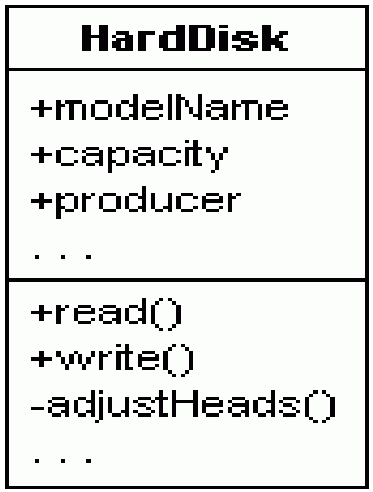
- Visibility: specifies the extent to which other classes can use a given class's attributes or operations.

Three levels of visibility:

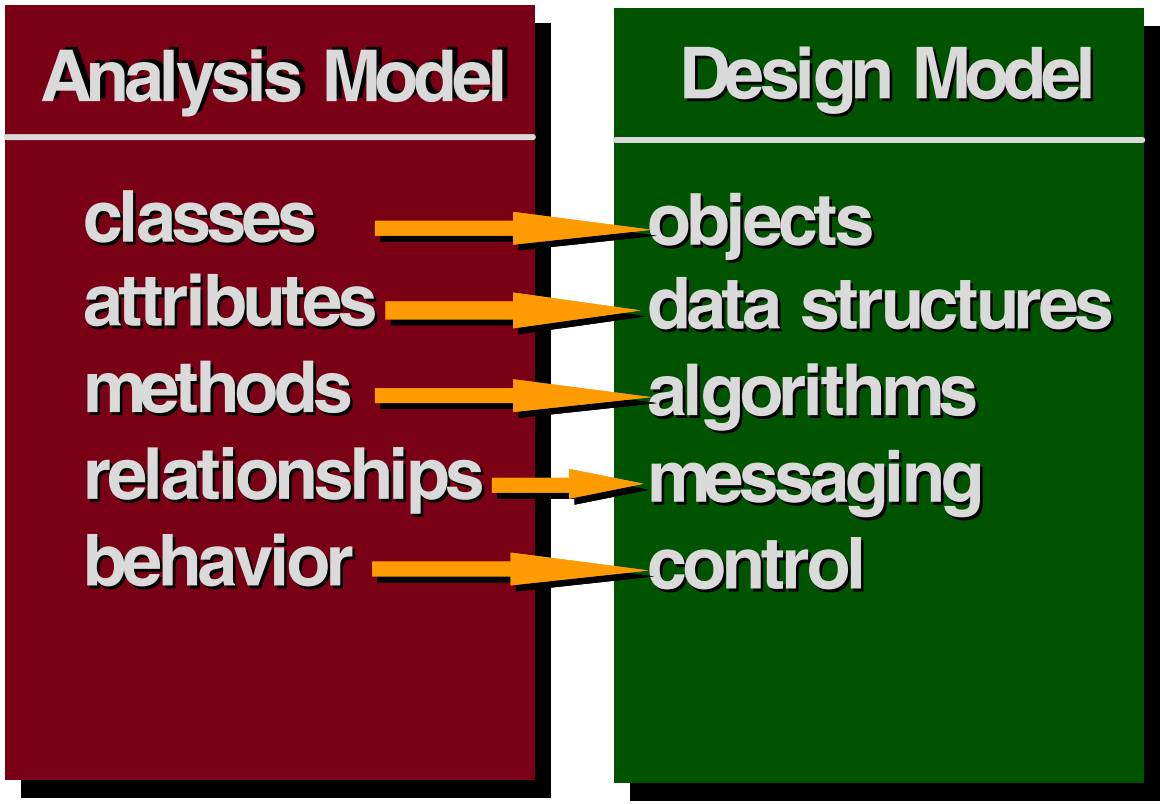
- **+ : public level** (usability extends to other classes)
- **# : protected level** (usability is open only to classes that inherit from original class)
- **- : private level** (only the original class can use the attribute or operation)

OO: Visibility

Ex: *Public and private operations in a Hard Disk*



OOA and OOD


Analysis Model
classes
attributes
methods
relationships
behavior
Design Model
objects
data structures
algorithms
messaging
control

Object behaviour modelling

- A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case

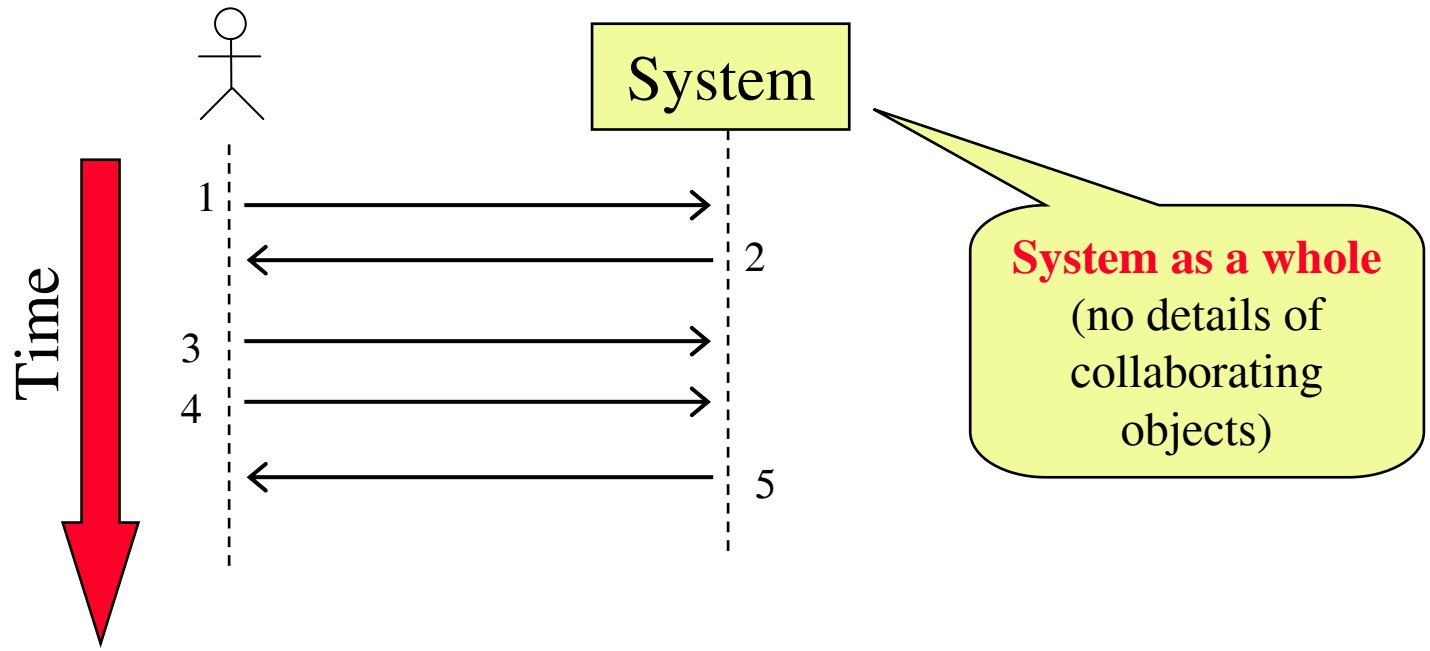
- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects

Sequence Diagram

- Class diagrams and object diagrams represent **static** information.
- In a functioning system, however, objects interact with one another, and these interactions occur over time.
- The UML sequence diagram shows the **time-based dynamics of the interaction**

System Sequence Diagram (SSD) for a UC

- For any UC Scenario (e.g. Happy Path)



Message type in a sequence diagram

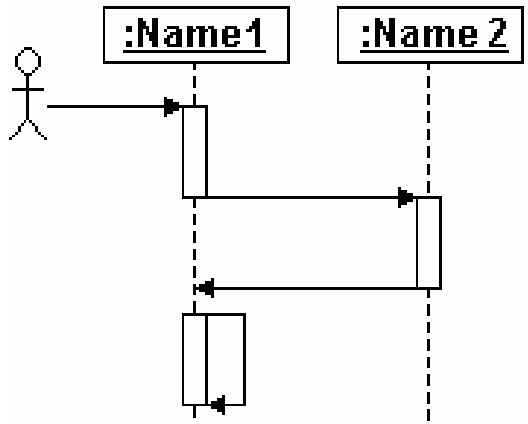
- ***Simple*** This is a transfer of control from one object to another.
- ***Synchronous*** If an object sends a synchronous message, it **waits** for an answer to that message before it proceeds with its business.
- ***Asynchronous*** If an object sends an asynchronous message, it **doesn't wait** for an answer before it proceeds.

Sequence Diagrams (SD)

- A SD represents time in the vertical direction.
  - Time starts at the top and progresses toward the bottom.
  - A message that's closer to the top occurs earlier in time than a message that's closer to the bottom.
- Horizontal direction shows the objects participating in the interaction
- More objects may be **created** during the interaction
- Objects may be **destroyed** when not needed any more

Sequence diagrams

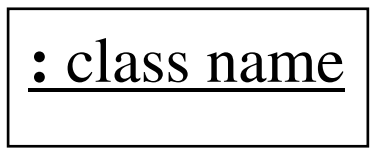
- The actor-symbol initiates the sequence,



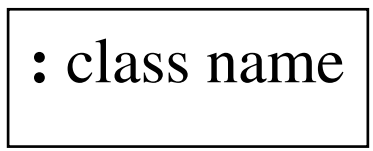
Object and class naming

- Named object

- Anonymous object (any object of the same class)

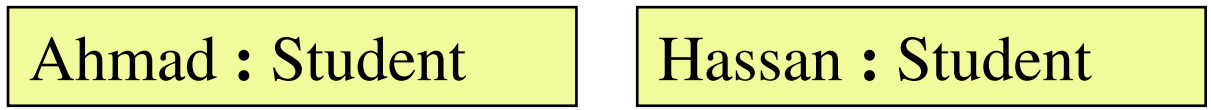


- Or less used

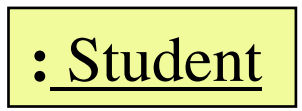


Object and class naming

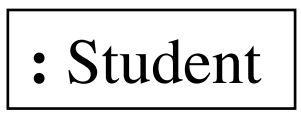
Named objects Ahmad and Hassan of class Student:



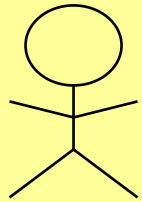
- Anonymous (any) student obj



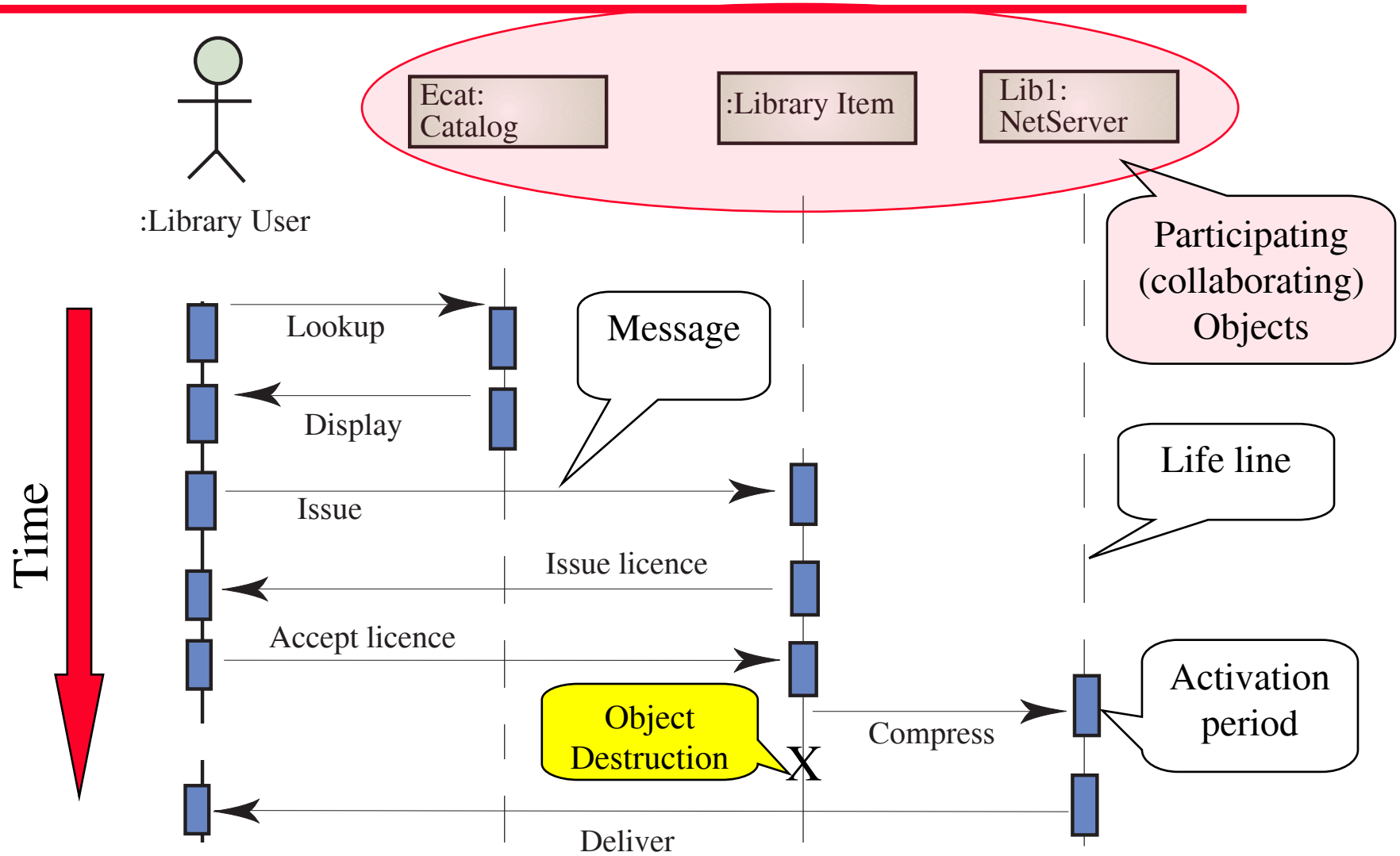
Or less used



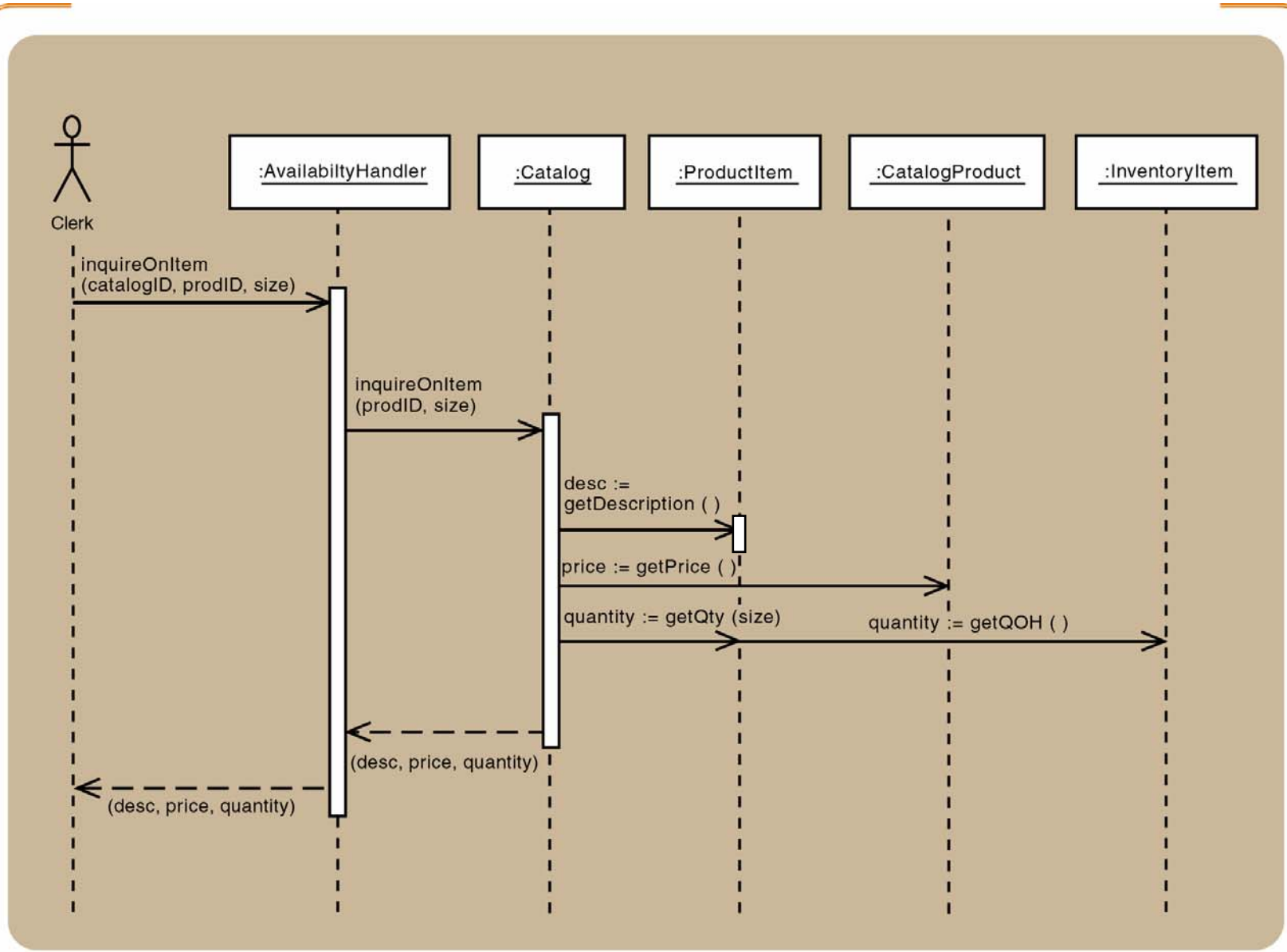
Sequence Diagram Syntax

AN ACTOR	
AN OBJECT	anObject:aClass
A LIFELINE	
A FOCUS OF CONTROL (Object activation)	
A MESSAGE	aMessage()
OBJECT DESTRUCTION	X

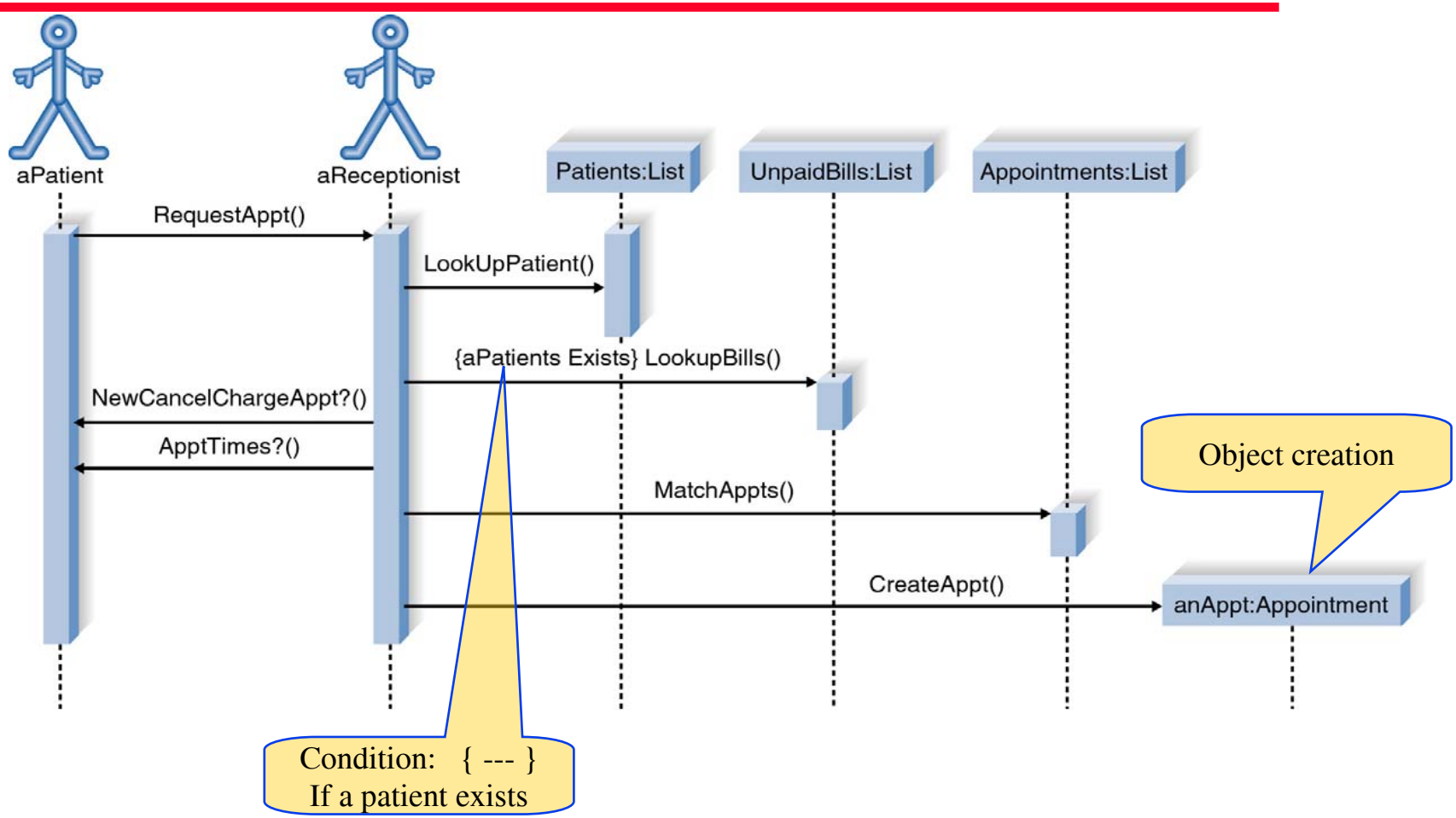
Sequence Diagram
Ex: Issue of electronic items



Sequence Diagram: Inquire Item



Request Appointment SD



Dennis: SAD
Fig: 8-1 W-30 100% of size
Fine Line Illustrations (516) 501-0400

Building a Sequence Diagram

1. Determine the **context** of the sequence diagram
2. Identify the **participating objects**
3. Set the lifeline for each object
4. Add **messages**
5. Place the focus of control on each object's lifeline
6. Validate the sequence diagram

Use Case & Sequence diagram

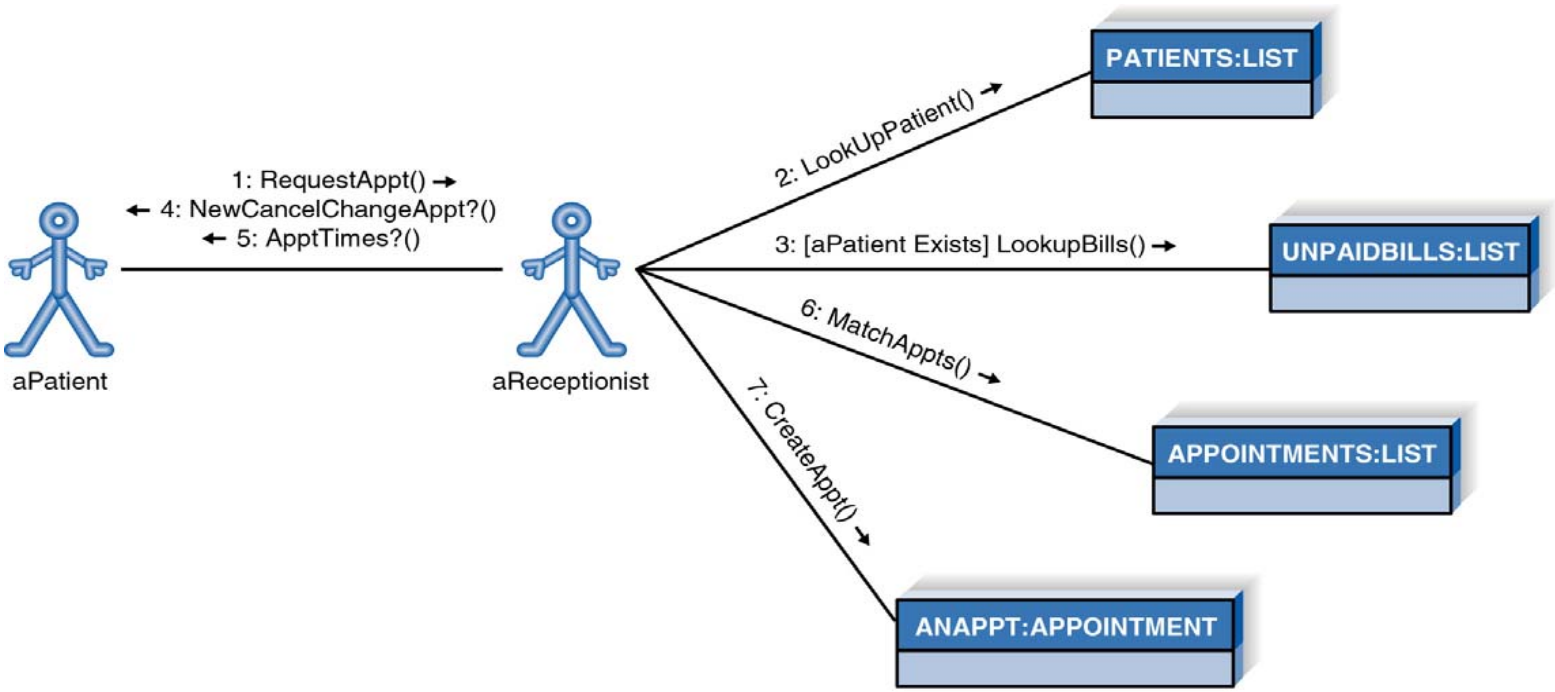
- How many SDs for one UC?
- One SD or more
- Build a SD for each scenario in the UC
  - Happy path scenario
  - Alternative scenario(s)
  - Exception scenario(s)

Communication 'UML 2.0' (Collaboration) Diagrams

- Object diagram that shows message passing relationships instead of aggregation or generalization associations.

- Emphasize the **flow of messages** among objects
  - Recall: SD emphasize timing of messages

Ex: Communication (Collaboration) Diagram



Dennis: SAD
Fig: 8-6 W-33 100% of size
Fine Line Illustrations (516) 501-0400

Elements of Communication (collaboration) Diagram

Actor	
Object	
Association	
Message	
Frame	

CRUD Analysis & CRUD matrix

- Interaction among objects (instances of classes)
- UC CRUD matrix:
  - for single Use Case
- System CRUD matrix

CRUD Analysis & CRUD matrix

	Customer	SearchReq	CDList	CD	Mkt Info	Review	Artist Info	Sample Clip	Shopping Cart	Order
Customer		R							U	C
SearchReq			CR							
CDList										
CD					R					
Mkt Info						U	U	U		
Review										
Artist Info										
Sample Clip										
Shopping Cart										
Order										

FIGURE 8-9 CRUD Matrix for the Place Order Use Case

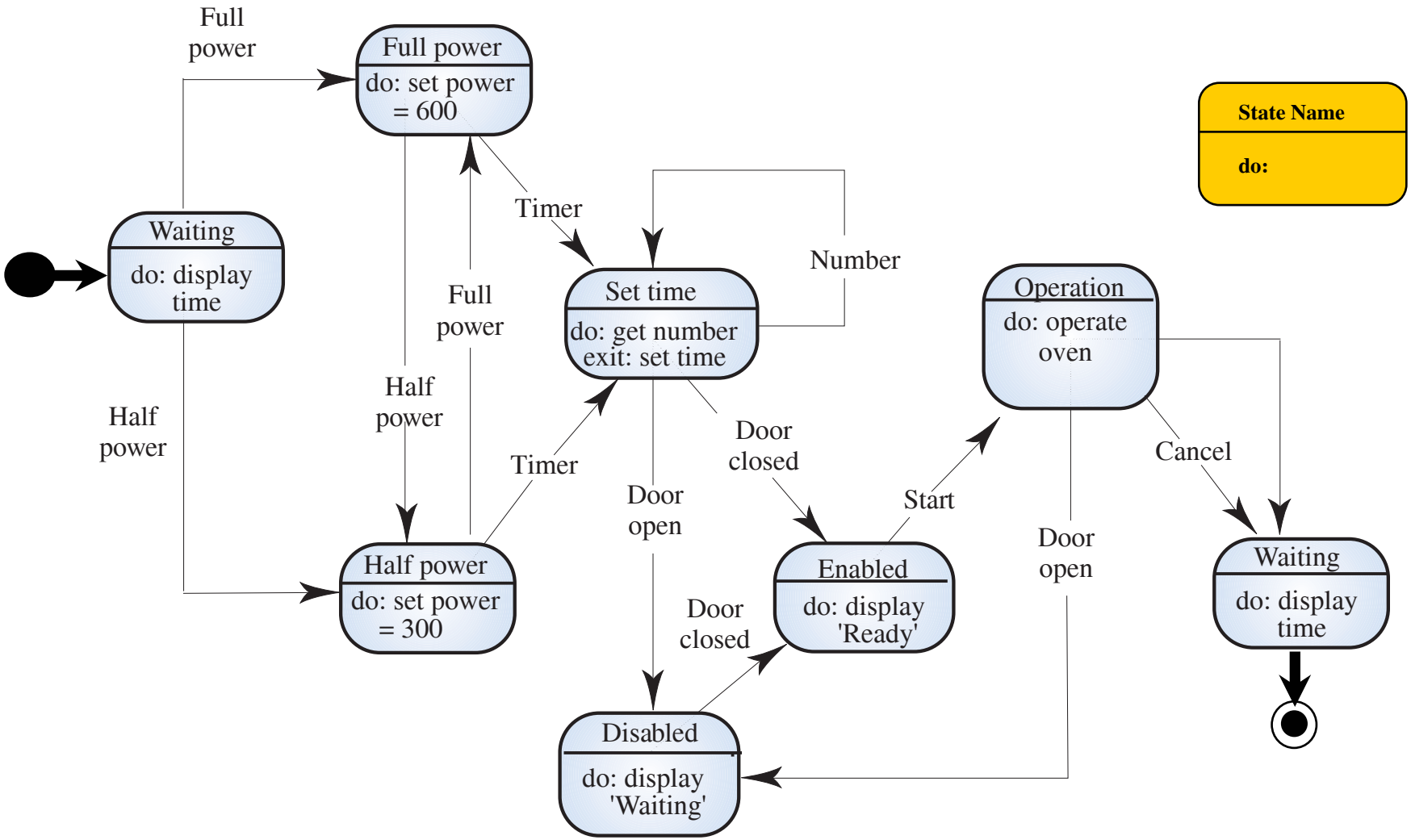
Building a Communication (Collaboration) Diagram

1. Determine the context of the diagram
2. Identify the participating objects and their associations
3. Layout objects and associations
4. Add messages
5. Validate the sequence diagram

State machine models

- These model the behaviour of the system in response to external and internal events
- They show the system's responses to stimuli so are often used for modelling **real-time** systems
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another
- Statecharts are an integral part of the UML

Microwave oven model



Statecharts

- Allow the decomposition of a model into sub-models (see following slide)
- A brief description of the actions is included following the ‘do’ in each state
- Can be complemented by tables describing the states and the stimuli

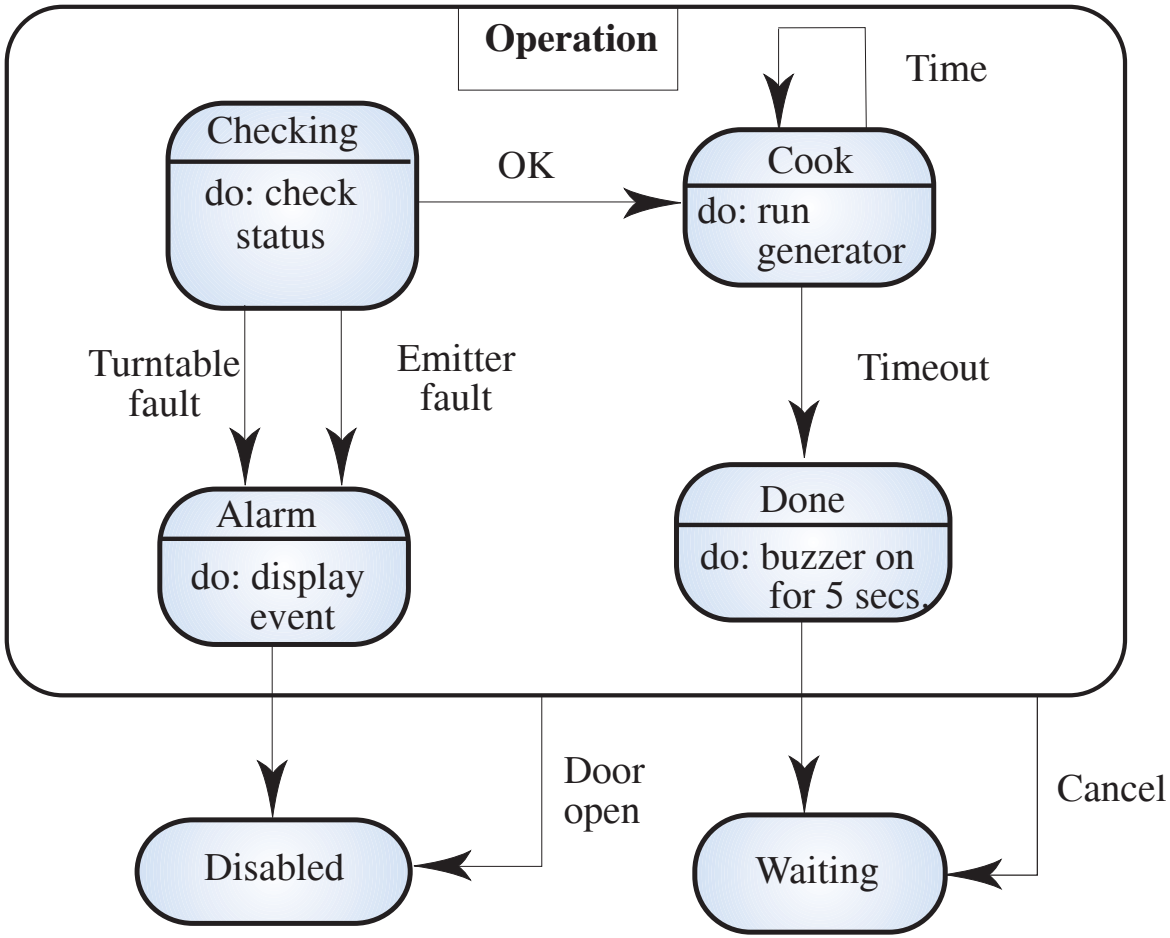
Microwave oven state description

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for 5 seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.

Microwave oven stimuli

Stimulus	Description
Half power	The user has pressed the half power button
Full power	The user has pressed the full power button
Timer	The user has pressed one of the timer buttons
Number	The user has pressed a numeric key
Door open	The oven door switch is not closed
Door closed	The oven door switch is closed
Start	The user has pressed the start button
Cancel	The user has pressed the cancel button

Microwave oven operation



UML: State Transition