UML Diagrams:
StateCharts
The Dynamic Analysis Model

Instructor: Dr. Hany H. Ammar
Dept. of Computer Science and Electrical Engineering, WVU
UML Development - Overview
The Requirements Model and the Analysis model
The Analysis Model and the importance Statecharts
Finite State Machines and Statecharts
More on State Chart Elements
Examples
The Requirements Model and the Analysis Model

Requirements Elicitation Process

The Analysis Process

- Functional/Nonfunctional Requirements
- Use Case Diagrams/Sequence Diagrams (the system level)

- Static Analysis
- Dynamic Analysis

- Class Diagrams
- State Chart Diagrams/Refined Sequence Diagrams (The object level)
outline

- UML Development - Overview
- The Requirements Model and the Analysis model
- The Analysis Model and the importance Statecharts
- Finite State Machines and Statecharts
- StateChart Elements
- Examples
The Analysis Model and the Importance of StateCharts

StateCharts are particularly important for real-time systems,
Control functions are typically activated at specific states of the system
StateCharts model the dynamic behavior of an object (with multiple states of behavior) by showing the possible states that the object can be in (idle, busy, waiting for selection, timedout, processing_transactions, etc)

In the analysis model a StateChart diagram is needed for each class of domain objects (including the system class) defined in the class diagram that has multiple states of behavior.
Recall the Banking System Example
Example: StateChart for class Account in an ATM example

![StateChart for class Account in an ATM example](image)

**Figure 10.1 Example of Account statechart**
Example: StateChart for the ATMControl class

What are the limitations of this Diagram?
UML Development - Overview
The Requirements Model and the Analysis model
The Analysis Model and the importance of Statecharts
Finite State Machines and Statecharts
StateChart Elements
Examples
Finite State Machines and Statecharts

- Statechart: Graphical representation of finite state machine—States are rounded boxes—Transitions are arcs
- Statechart relates events and states of a class of objects
  - Event—Causes change of state Referred to as state transition
  - State—A recognizable situation—Exists over an interval of time—Represents an interval between successive events
Finite State Machines (FSMs) and StateCharts (SCs)

- SCs are graphical representation of FSMs
- They can depict complex FSMs consisting of a hierarchy of state diagrams
- SCs consist of states and transitions
- A state depicts an actual state of behavior that an object can be in during its life time
- A transition from one state to another is caused by an event (e.g., user input, received a message from another object, etc)
StateChart Rules
States of an Object

- A recognizable situation
- Exists over an interval of time
- Represents an interval between successive events
- Can be a Macro state or a Micro state
- A Macro State is defined by another StateChart containing Macro and Micro states
- A Micro state is a primitive state not defined any Further
Figure 4-3: Basic Statecharts
Figure 3-1: State Machine for an Object
Figure 4-10: Fork and Join
Figure 3-8: Forks and Joins
The History mark means subsequent entries are to the last active state.
outline

- UML Development - Overview
- The Requirements Model and the Analysis model
- The Analysis Model and the importance of Statecharts
- Finite State Machines and Statecharts
- More on State Chart Elements
  - Events, Conditions, Actions, and Activities
- Examples
Events

- Event–A discrete signal that happens at a point in time
  - Also known as a stimulus
  - Has no duration

- Two events
  - May logically depend on each other
  - E.g., ATM Card inserted before Pin # entered

- Two events
  - May be independent of each other (they can occur independently)
  - E.g., Cancel
Events and Conditions

- State transition label
  - Event [Condition]

- Condition is a Boolean function
  - Conditions are optional on statecharts
  - Condition is true for finite period of time

- When event occurs, condition must be *true* for state transition to occur.
Figure 3-6: Branches and Junctions
Actions

- Can be defined as state transition label
  - Event / action(s)
  - Event [condition] / action(s)

- Actions
  - Executed as a result of state transition
  - Executes instantaneously at state transition
  - Terminates itself

- Entry Actions
  - Defined for a given state and executes on entry to this state from any state

- Exit Actions
  - Defined for a given state and executes on exit from this state to any state
Example: Actions and Conditions, Auto Cruise Controller

**Figure 10.9** Detailed Cruise Control statechart with actions and conditions
Example: Entry Actions, execute on the entry to a state after a state transition

Figure 10.11b  Entry actions

Figure 10.11  Example of entry actions (continued)
Example: Exit Actions

Figure 10.12a  Actions on state transitions

Figure 10.12b  Exit action
Activities

Activity
- Executes for duration of state
  - Enable Activity on entry to state
  - Disable Activity on exit from state

Examples of activities
- Increase Speed
  - Executes for duration of Accelerating state
- Maintain Speed
  - Executes for duration of Cruising state
- Resume Cruising
  - Executes for duration of Resuming state
Example: StateChart with Activities, Auto Cruise Controller

Figure 10.10  Cruise Control statechart with activities
UML Development - Overview
The Requirements Model and the Analysis model
The Analysis Model and the importance of Statecharts
Finite State Machines and Statecharts
StateChart Elements
Examples
The Sound Recorder
Analysis Level Class Diagram

Figure 3.2: Sound Recorder class diagram
Figure 4.2: AudioInput statechart

Figure 4.3: AudioOutput statechart
Incomplete Statechart and incorrect state label

Figure 4.1: Behaviour of an AudioController

Why?
Example: Digital Sound Recorder
User Interface subsystem Design Class diagram

Figure 3.11: User interface subsystem class diagram
Example: Digital Sound Recorder
StateChart of MenuUserMode class

Figure 4.7: MenuUserMode statechart
Figure 4.8: SettingClockUserMode statechart
Figure 4.9: SettingDateUserMode statechart
Recall the ATMControl class

**Figure 21.11.** Communication diagram: ATM client Validate PIN use case

PIN Validation Transaction = \{transactionId, transactionType, cardId, PIN, startDate, endDate\}
Example: StateChart for the ATMControl class

What are the limitations of this Diagram?

Figure 10.2  Example of flat ATM statechart
Example: Macro States, Hierarchical StateCharts ATM system

Figure 10.14 Example of hierarchical statechart
Example: Auto Cruise Control and Monitoring (The Cruise Cont. Subsys)
Example: Auto Cruise Controller initial statechart

**Figure 10.9** Detailed Cruise Control statechart with actions and conditions
Example: Auto Cruise Controller Refined statechart

![Statechart Diagram]

Figure 10.19 Hierarchical Cruise Control statechart with activities and exit action
Example: Elevator Control Collaboration Diagram

- **Display**: Display
- **Door**: Door
- **Floor Sensor**: Floor Sensor
- **Central Station**: Central Station
- **Inside Elevator Buttons**: Inside Elevator Buttons

**Floor_No_Arrived**

1: state “Idle”, Floor_No_Arrived=1
3: state “Going_Up”, Req_Highest_Floor=7
4: state “Passenger_Loading”, Floor_No_Arrived=7
8: state “Going_Down”, Req_Lowest_Floor=5
9: state “Passenger_Loading”, Floor_No_Arrived=5
12: state “Idle”, Floor_No_Arrived=5

2: Req_Floor=7
5: Req_Floor=5
6: Open_Door
10: Open_Door
7: Door_Closed=TRUE
11: Door_Closed=TRUE
Scenario of the Collaboration Diagram

• Idle on Floor 1
• Gets request from Floor 7
• Going Up to Floor 7
• Gets request from inside passenger to Floor 5
• Loading on Floor 7
• Going Down to Floor 5
• Loading on Floor 5
• Idle on Floor 5
Elevator SW Sys
State Diagram

Out_of_Service

Alarm is off

In_Service

Req_Floor=NULL

Floor_No_Arrived>Req_Floor

Floor_No_Arrived=Req_Floor

Open_Door

Floor_No_Arrived!=NULL

Req_Floor!=NULL

Passenger_Loading

Floor_No_Arrived<Req_Floor

Floor_No_Arrived<Req_Highest_Floor

&& Door-Closed=TRUE

Floor_No_Arrived > Req_Lowest_Floor

&& Door-Closed=TRUE

Floor_No_Arrived <= Req_Floor

Open_Door

Floor_No_Arrived > Req_Floor

Open_Door

Going_Up

Floor_No_Arrived < Req_Highest_Floor

&& Floor_No_Arrived != Req_Floor

Floor_No_Arrived < Req_Highest_Floor

&& Door-Closed=TRUE

Floor_No_Arrived != Req_Floor

Open_Door

Floor_No_Arrived > Req_Lowest_Floor

&& Door-Closed=TRUE

Floor_No_Arrived > Req_Floor

Open_Door

Going_Down

Floor_No_Arrived > Req_Lowest_Floor

&& Door-Closed=TRUE

Floor_No_Arrived != Req_Floor

Open_Door

Floor_No_Arrived > Req_Floor

Open_Door

Alarm is on
Example: Elevator Control, a better example

Figure 18.4: Elevator Control System context class diagram.
Figure 18.13 Hierarchical statechart for Elevator Control
Example: The Pacemaker

There are two queues, one for sending and one for receiving, just to simplify life.

Figure 4-15: Pacemaker Class Diagram
Figure 4-18: Communication Gnome State Model
Figure 4-19: Chamber Model State Model
Figure 4-20: Atrial Model State Model
Figure 4-21: Ventricular Model State Model