Modeling Event-Based Behavior with State Machines

Critical Embedded Systems

*Dr. Ákos Horváth and Dr. Balázs Polgár*

*Prepared by*
Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Dept. of Measurement and Information Systems

© All rights reserved.

This material can only be used by participants of the course.
System Modeling Process

1. Collect Textual Requirements
2. Model System Context
3. Model Functional Requirements with Use Cases
4. Elaborate Functional Requirements with Activities
5. Create Data Model
6. Model Interactions between System and Environment
7. Derive System State Model
8. Derive System Interfaces
9. Model System Structures
What is it about?

Context of the Modeling Aspect
State Machine Diagram
What are the states of the selected component?
How it reacts to events (how it changes states)?
What are the building blocks?

Modeling Elements & Notation
Event:

- Asynchronous occurrence/happening with parameters e.g. mouse click and its place and which button
- Full-fledged object, instance of the Event class inheritance: extension of its attributes
- Life-cycle:
  - Initialization, notification of target objects
  - Event-queues and selection
  - Processing
- Reactive objects: react to events
Atoms of dynamic modeling II.

- **Operation:**
  - Services provided by the classes (methods)
    - client-server relation
    - can have return values
  - *Part of the class definition*
  - *Synchronous or asynchronous communication* between objects
    - e.g., method invocation
      
      
      ```
      result = server->operation(p1, p2, ..., pn)
      ```

- **Signal reception**
  - *Asynchronous communication* between objects
**State:**
- The state of an object
- Defined by:
  - value of its attributes (e.g., $x<3$)
  - conditions are met (e.g., operation can be executed)

**Transition:**
- Change of state
- Triggered either by the incoming event or completion

**Action:**
- The operations to be executed by the object
Dynamic Modeling

with State Machines
State Machines

- Describes the states and state transitions of the system, of a subsystem, or of one specific object.
  - hierarchical and concurrent systems

States
- Concrete state:
  - Combination of possible values of attributes
  - Can be infinite
- Abstract states: (like in State Machines)
  - Predicates over concrete states
  - One abstract state \(\mapsto\) many concrete states
  - Hierarchical states:
    - Frequent in embedded apps (e.g. control of car brake)

Transitions
- Triggering Event
- Guard
- Action
State Machine - introduction

- For defining reactive behavior of objects
  - Responds to events: state transitions and actions
  - Traditional approach: state machine

- UML State Machine: extension to state machine
  - **State hierarchy**: refinement of states
  - **Concurrent behavior**: parallel threads
  - **Memory**: last active state configuration
States I.

- **Attributes:**
  - entry action
  - do action
  - exit action

- **State refinement**
  - *Simple state*
  - *OR refinement*: auxiliary state machine, only one active state
  - *AND refinement*: concurrent regions (state machines), all regions are active in parallel
Example: State refinement I.
Example: State refinement II.
Example: State refinement III.

AND refinement
Example: State refinement IV.

On

Image

On

Sound

SoundOn

 SND

mute

SoundOff

Off

Standby

Disconnected

off

on

out

in

coordinates

dimensions: 720.0x540.0

[Image 0x1 to 100x29]

[Image 633x-0 to 717x28]

[115x406]

[511x406]

[553x208]

[529x208]

[85x382]

[283x328]

[289x184]

[295x262]

[355x262]

[565x262]

[631x262]

[439x352]

[439x214]

[445x298]

[127x340]

[115x160]

[121x202]

[121x496]

Example: State refinement IV.
Example: State refinement V.
State II.

- **History state**
  - Stores the last active state configuration
  - Input transition: it sets the object to the saved state configuration
  - Output transition: defines the default state, if there were no active state since
  - Deep history state: saves the complete state hierarchy (down to the lowest substates)

- **Initial state:** becomes active when entered to the region
  - One in each OR refinement
  - One in each AND region

- **Final state:** state machine terminates
Example: History State

```
Example: History State

Print_job

Handle
  Get
  Reply

Process
  Print
  Close

it

H
```

Transition I.

- Defining state changes

- Syntax:

  ```
  trigger [guard] / action
  ```

  - **trigger**: event, triggered operation or time-out
  - **guard**: transition condition
    - Logic formula over the attributes of the objects and events
    - Referring to a state: IS_IN(state) macro
    - Without trigger: if becomes true the transition is active
  - **action**: operations $\Rightarrow$ action semantics
Transition II.

- **Time-out trigger:**
  - becomes active if the object stays in the source state for the predefined interval
    - e.g., tm(50), based on system time

- **Complex transitions**
  - **Fork**
  - **Join**
  - **Condition**
  - **(Internal)**
    - executes without exiting or re-entering the state in which it is defined

- **Transitions between different hierarchy levels**
Transition example

- **Prepare**
  - **Group1**
    - **Phase1**
    - **Act1**
  - **Group2**
    - **Phase2**
    - **Act2**
  - **Act3**

- **Work**
- **State name**
- **Passed**
- **Missed**
- **tm(50)**

- **Illegal activity**
  - **[fatal] / report_status()**
  - **[not_fatal] / recovery()**

- **Error**

- **Failure**
(Basic) State Machine elements

- State
- (Transition)
- History state
- Initial State
- Final State
- Conditional transition
- Synchronization (fork/join)
How is the model interpreted?

Semantics of the Model
Semantics: How does it work?

- Basics:
  - Hierarchical state machine (state chart)
  - Event queue + scheduler

- Semantics defines:
  Behavior in case an event occurs
  \[\rightarrow\] one step of the state chart
  - (concurrent) transitions fire
  - State configuration changes
    in all region in the active state and also one substate in the OR refinement (recursively)
Semantics of State Transitions

- Separately processed events:
  - Scheduler only triggers the next event if the previous one is completely processed
    - stable configuration: there is no state change without an event

- Complete processing of events:
  - The largest set of possible fireable transitions
    - all enabled transition fires, if they are not in conflict

How does it work?: Steps of the event processing
Steps of event processing I.

- Scheduler triggers an event for the State Machine in a stable state configuration

- Enabled transitions:
  - Source state is active
  - The event is their trigger
  - Guards are evaluated to true

Based on the number of fireable transitions
  - Only one: fire!
  - None: do nothing
  - More than one: select transitions to fire?
All transitions are triggered by the same e event: Which should fire?

Disabled (cannot fire): t5
Cannot fire together : (t1,t2); (t1,t4); (t2,t4); (t3,t4)
Steps of event processing II.

- Selection of fireable transitions:
  - Fireable = Enabled + Max, priority
  - Conflict: Has the same source state
    - Formally: the intersection of their left (exit) states is not empty
  → Conflict resolution → priority:
    - Defined between two transitions (t₁ and t₂)
    - t₁ > t₂, if and only if the source state of t₁ is a substate within the state hierarchy of t₂ ("lower level")
Steps of event processing III.

- Selection of transitions to fire:
  - Set of transitions to fire: parallel execution of concurrent transitions:
    - Maximum number of fireable transitions (= cannot be extended any further)
    - There is no conflict between any two transitions
  - Selection of this set:
    - Random!
Conflict resolution

Fireable: (t1,t3) or (t2,t3)
Steps of event processing IV.

- Selected transitions fire:
  - in random order

- Firing one transition:
  - Leaving the source states from the bottom to top and execute all their *exit* operations
  - Execute the action of the transition
  - Entering the target states from top to bottom and execute the *entry* actions \(\rightarrow\) new state configuration
Steps of event processing V.

- Entering a new state configuration:
  - Simple target state: part of the state configuration
  - Non-concurrent superstate: direct target of one of its substate or its initial state
  - Concurrent target state: all of its regions have to have an active state either as direct target state or with initial state
  - History state: the last active state configuration if there is none: the target state of the history state
State transition example
State transition example

S11
- S111
  - S1111
  - S1112
  - S1113

S112

S12
- S121
  - S1211
  - S1212

S122

S123
- S1231
- S1232

S1211 - exit action
State transition example

S11
- S111
  - S1111
  - S1112
  - S1113

S12
- S121
  - S1211
  - S1212
- S122
- S123
  - S1231
  - S1232

S121 - exit action
State transition example
State transition example
State transition example
State transition example

S11

S111

S1111

S112

S1112

S1113

S112

S111

S12

S121

S1211

S1212

S122

S123

S1231

S1232

S111 – entry action
State transition example
Summary

- Effective technique to model certain dynamic systems

- Hierarchic refinement allows iterative development

- Already used in many application domain
  - Avionics, automotive, control, etc.
Complex Example

- Traffic light for an intersection with a prioritized road
  - Off: (blinking yellow)
  - On: green for the priority road
  - Green, yellow, red etc. Different timerange (timer)
  - 3 waiting vehicle on priority road: green light despite the timer’s ticks
  - Automatically take photos of vehicles crossing the priority road on red light. Manual on/off for this feature.
1. Basic state machines

- Off
- do/blink
- Reset
- Red
- Yellow
- Green
- Red
- Yellow
- T1
- T2
- T3
- T4
- tm(T4)
2. Hierarchy

- Off
  - do/blink
  - !reset

- On
  - T1
  - T2
  - Green
    - T4
  - Yellow
  - Red
  - Yellow
  - Red

- Red

- Reset
  - !reset

Diagram shows a flowchart with states and transitions between Off and On, and color states Red, Yellow, and Green, with specific transitions labeled T1, T2, T3, and T4.
3. Concurrent states

- **Off**
  - do/blink

- **On**
  - T2
  - Yellow
  - T1
  - Green
  - T4
  - Red
  - Yellow
  - T3

- **Red**
  - Camera
  - Count

- reset

- !reset
4. History States

Off
  do/blink
  \[!\text{reset}\]

On
  T2
  Yellow
  T1
  Green
  T4
  Red
  Yellow
  T3

Red

Camera

Count

CarGo

H

Shoot

Manual

Manual Off
Complete System

Off

\( do/\text{blink} \)

reset

\( \text{reset} \)

On

T2

Yellow

Green

Red

Yellow

Off

\( \text{T1} \)

\( \text{T2} \)

\( \text{T4} \)

\( \text{T3} \)

Red

Count

Count0

CarGo

Shoot

ManualOn

ManualOff

Camera

Count1

Count2

\( \text{car} \)

\( \text{car} \)

\( \text{car} \)