3.6 State Diagrams

Subject/Topic/Focus:
- Introduction to State Diagrams

Summary:
- States
- Transitions
- Events
- Composition
- Concurrency
- Branch
- History

Literature:
- [Fowler99]
- [Booch98]

State Machines in UML

A state machine
- specifies state changes which an object performs triggered by events or signals received from other objects,
- specifies the reactions of the object on events,
- specifies actions in states,
- defines protocols, i.e., legal sequences of operation calls of a class or interface or of interactions by signals.

- is attached to exactly one class
- can be inherited by subclasses
  (where the concept of the refinement is not defined exactly)
States and State Diagrams

A state diagram is a graph consisting of

- **states**
  - simple states
  - composite states (simple states refined by nested state diagrams)
- **state transitions** connecting the states.

A state is a constraint or a situation in the life cycle of an object, in which a constraint holds, the object executes an activity or waits for an event.

Notation:

![State Diagram Notation](image)

Start and Final States

Start state: State transition is executed immediately during the creation of the object.
Only possible event: create(parameter)
Java: constructor (new)

Notation:

![Start and Final States Notation](image)
Transitions

A transition connects two states and shows the flow of control. A transition can include a triggering event, a guard and actions to be executed. Transitions without event and guard are executed immediately when an activity is finished respectively all sub states were passed through.

Notation:

```
login(user) [NumberOfUsers < max] / greet(user)
```

```
login(user) [NumberOfUsers = max] / deny(user)
```

Events

An event is a phenomenon in space and time significant for the modeled system.

An event can appear synchronously or asynchronously.

- **synchronous** events:
  - call event: triggered by call
  - exception event: triggered by called object at return

- **asynchronous** events:
  - signal event: signal sent by other object
  - change event: triggered by side effects on object attributes
  - time event: spontaneously triggered by boolean guard over time

An event can trigger state changes.
Signal Events and Call Events

Signals are asynchronous, i.e., the sender does not wait until the receiver received the signal or reacted on it.

A call event is triggered by a (synchronous) operation call.

Call events are synchronous, i.e., the sender waits until the receiver reacted on the event.

In the state automaton signals and call events are indistinguishable from each other.

The receiver can

- ignore the event (the event is lost),
- execute its trigger event or
- execute an operation.

Call Events

**Notation:**

- `event`
- `parameter`
- `state`
- `manual control`
- `automatic control`
- `startAutopilot(goal)`

Name and parameter of the event must be compatible to methods of the class.
Time and Change Events

A **time event** appears after the expiration of a time period.

A **change event** occurs if a specific constraint is fulfilled. The constraint is a boolean expression on the attributes of the actual object.

**Notation:**

```
when NumberOfUsers = 0
```

```
active after 10 minutes backup
```

```
change event
```

Sending Signals

Signals can be sent to other objects during a transition.

**Notation:**

```
active after 10 min. / send central.amAlive(self)
```

```
send signal
```

```
alternative
```

```
active after 10 min
```

```
central
```

```
receiving object
```
Triggering Actions

Possible actions:
- send signal
- perform call
- perform access

Notation:

- entry action
- exit action
- intern transition
- activity
- delayed event

Notation:

Entry action:

- active
  - entry / startMotor(3000 U/Min)
  - exit / stopMotor()
  - new speed x / speed(x)
  - do / warningLight()
  - turn electricity off / defer

Composite States

A state can be refined hierarchically by composite states.

Notation:

- composite state
- sub state

Transition from each sub state

- wait
  - card inserted
  - abort
- ready

Transition from sub state

- key in the machine
  - card inserted

Transition in sub state

- start state
  - inspection
  - choice
  - transaction

Entry / read card
Exit / eject card
Concurrent Sub States

In a state several sequences of sub states described by own state machines can be performed concurrently.

Notation:

- **fork**
- **finish**
- **error**
- **join**

**final test**
- hardware test
  - test ports
  - test devices

- software test
  - test modules
  - test application

**concurrent sub state**

Alternative: active objects

**Concurrent Sub States: Alternative**

- **fork**
- **join**

**final test**
- hardware test
  - test ports
  - test devices

- software test
  - test modules
  - test application

**installation**

**operating**