Behavioral Requirements

Behavioral Models

Decision-Oriented BM

State-Oriented BM

- Finite State Machines
- Protocol Validation & Verification
- Statecharts
- Petri Nets

Function-Oriented BM

Expressive Power Revisited:

"Can we precisely specify behavior using OO-methodology?"

[BOF session, by Booch & Rumbaugh, OOPSLA, 94]

"Q: ... I don’t see any methods providing something for precise specification of behavior."

[... Booch turns the question around ...]

"Q: ... Well, again, I’m not asking to precisely specify everything. I’m asking for the method to give me some way to precisely specify something when I need to precisely specify. Otherwise the programmer will not know what I’m saying."

[Rumbaugh: I am not a great believer in formal languages ...]

"Q: ... I’m asking for some way to precisely specify behavior."

Finally, Grady Booch answered:

"I understand. Short answer is we’re getting defensive here, Hmm... Short answer is look at our pre- and post-conditions."
**Behavioral Models**

- Behavioral models offer facilities for modelling the behavior of a process or activity

- Can be classified into:

  - **decision-oriented models:** view a process as an algorithm for deciding the outcome of an event, given some set of inputs
    - e.g., decision on a loan application, given info. about the applicant
    - e.g., decision tables/trees

  - **state-oriented models:** view a process as a collection of "states" and "transitions" among those states
    - e.g., Finite State Machines (FSMs), StateCharts, Petri Nets

  - **function-oriented models:** define a process in terms of a set of "preconditions" which must be true before the process is initiated, and "postconditions" which must be true after the process is completed.

**Decision-Oriented Behavioral Models**

*How do we specify what a process does ("process logic")? How do we represent policies?*

- Decision tables enumerate parameters (or "conditions") and "actions" (or "outcomes") that should take.

> "If the plane is more than half full and the flight costs more than $350 per seat serve free cocktails, unless it is a domestic flight. Charge for cocktails in all domestic flights where cocktails are served, i.e., those that are more than full."

<table>
<thead>
<tr>
<th>Domestic</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
<th>N</th>
<th>N</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;= half full</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>&gt;= $350/seat</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

| Serve cocktails | X | X | X | ? | ? | ? |
| Free cocktails   |    |   |   | X |

1 Each condition is usually \( T, F, \) don’t-care; but can also be multi-valued \{ never, rarely, sometimes, often, always \}
1 total # of columns = total # of rules = \( X \) (# input-cond)
State-Oriented Behavioral Models

- describe a system in terms of states and transitions
- a state represents the static properties of a system at a particular time; states may have a duration and may have associated invariants.
  e.g., patient-waiting (in a doctor’s office) may have
  as duration the length of the wait and
  invariant that the person waiting wants to see the doctor.
- a transition represents a dynamic change of the system being modelled from one state to another; transitions may have triggers and associated actions
  e.g., the see-doctor transition changes the state of the system from patient-waiting to patient-examined

Finite State Machines (FSMs)

- A finite state machine (FSM) is an abstract machine which can be in exactly one state at any given time;
- While in a certain state Sc, a FSM accepts an input (I) and produces an output (O), while at the same time it changes state to Sn;
- Both output and next state depend only on current state and input:
  \[ S_n = F(S_c, I) \]
  \[ O = G(S_c, I) \]
- Two common notations for defining FSMs:
  state transition diagrams (STDs) and state transition tables (STTs)

STDs:

Example:

[Diagram showing state transitions with inputs and outputs, such as dial, on hook/make quiet, distinct dial tone, busy, etc.]
Finite State Machines (FSMs)

### STDs:
- **State**: idle, dial tone, on hook, make quiet
- **Transition**: caller off hook, gen dial tone, dial busy number, gen busy tone
- **Input/Output**: dial tone
- **Stimulus/Response**: distinct dial tone, dial 9, gen distinctive dial tone

### STTs:

<table>
<thead>
<tr>
<th>States</th>
<th>Stimuli</th>
<th>Caller Off Hook</th>
<th>On Hook</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tones</td>
<td>next state</td>
<td>dial tone</td>
<td>idle</td>
</tr>
<tr>
<td></td>
<td>response</td>
<td>gen dial tone</td>
<td>make quiet</td>
</tr>
</tbody>
</table>

**Example:**
- idle dial tone -> on hook/make quiet
- dial tone -> ring back tone
- dial idle number/gen ring back tone
- busy
dial busy number/gen busy tone
- gen ring back tone

Lawrence Chung

Computer Science Program, The University of Texas, Dallas

Protocol Validation and Verification

*Using Finite State Machine*

*What, Why & How*
Protocol Validation & Verification: **What?**

- **Specification** → **Real Needs**
  - **Validation:** ensure that a protocol specification is equivalent to real needs, or at least, logically consistent.
  - **Verification (Conformance Test):** ensure that the external behavior of a protocol implementation is equivalent to its specification.

Protocol Validation & Verification: **Why?**

- **Complexity:**
  Communication Protocols in distributed systems and open communication networks are highly complex.

  *But,* an ad hoc approach can make the problem more complex.

- **Error-prone:**
  Communication Protocols often contain errors. Specifications are ambiguous, incomplete, inconsistent, ...

  *But,* an informal approach has its limitation.

After all, what good is an international standard that is incomplete or even faulty?
**General Types of Protocol Errors:**

- **Deadlock:**
  A global state reachable from initial global state, in which no transmissions are possible.

- **Non-executable Interaction:**
  A reception/transmission that is specified in the design but never executed.

- **Unspecified Reception:**
  A reception that is executable but not specified in the design.

- **Overflow:**
  A channel state in which the number of messages exceeds some predefined bound.

---

**Protocol Validation & Verification: How?**

*The Finite State Machine Approach*

Example: How to model communication protocols using finite state machine?

**Process 1 (Client)**

1. Ready
2. Wait
3. Register
4. Service

**Process 2 (Server)**

1. Idle
2. Fault
3. Service

Channels:
- : transmission
- +: reception

A simple protocol between 2 processes: **Client & Server**
Error Detection:  How?

The Finite State Machine Approach

Example:

Process 1

1
-1
+2
3

Process 2

1
-3
+1

deadlock:

1 2 3

non-executable
interaction:

1 2 3

unspecified
reception:

1 3

Lawrence Chung

Validation & Verification Techniques:

• Reachability Analysis:
  
  ( Full Search
  ( Controlled Partial Search
  ( Random Search

• Conformance Test:
  
  test for every state and input that the protocol outputs according to specification.
  
  ( Generate conformance test suite
  ( Apply test suite to implementation

Of interest to international standardization bodies

Lawrence Chung
Statecharts

- A generalization of FSM notations, intended to facilitate the definition of large FSMs.
- Defined in terms of states and transitions too.
- However, more than one state may be “on” at any one time (for concurrency).
- Moreover, statecharts consist of several FSMs composed in terms of “AND” or “OR” compositions.

Conditional transition: transition from state S1 to S2 is to take place if there is input i and condition C is true.

\[ i(C) / O \]

Example:

```
S1
  dial tone
  gen ring back tone
  dial local number (callee idle)/gen busy tone
S2
```

```
\text{in FSM:}
\text{dial off hook/gen dial tone}
\text{dial idle number/gen ring back tone}
\text{dial busy number/gen busy tone}
```

Superstates: aggregate sets of states with common transitions.

\[ S1 \]

\[ \begin{array}{c}
S11 \\
S12 \\
\end{array} \]

\[ i(C)/O \]

Example:

```
idle
caller off hook/caller active
caller on hook
```

```
S1
  idle
caller off hook/caller active
caller on hook
dial tone
dial local number (callee idle)/gen busy tone
dial busy number (callee active)/gen busy tone
S2
```

Lawrence Chung
Statecharts

OR decomposition: One of the subordinate states is "on"

default entry state

AND decomposition: All of the subordinate states are "on"

"orthogonal" decomposition: state machines S1 and S2 are independent

Lawrence Chung

Petri Nets

A form of FSMs, first proposed by C. A. Petri in 1962

A notation for defining abstract concurrent processes

Primitives

- place
- transition
- token

When all input places of a transition are enabled (i.e., with a token) and the external stimulus associated with the transition occurs, the tokens move from the input places to output places

Example:

BEFORE

AFTER

Lawrence Chung
Fundamental notions:

- sequencing
  
  ![Petri Net diagram for sequencing]

- concurrency
  
  ![Petri Net diagram for concurrency]

- synchronization
  
  ![Petri Net diagram for synchronization]

- decision making
  
  ![Petri Net diagram for decision making]

Example

Stimuli occur in the following order: $T_1, T_3, T_5, T_4, T_6$
Augmented Petri Net

- **activation conditions and actions**
  
  e.g., for a patient waiting to see a doctor, an activation condition may be "doctor ready" and an associated action may be "patient move to doctor's office"

- **invariants on states:**
  
  e.g., "patient not feeling well"

- **communication primitives:**
  
  actions may involve communication with other Petri nets; e.g., "finding out test results" by communicating with the "labTests" Petri net

![Petri Net Diagram]

Function-Oriented Behavioral Models

- **describe what the process does, often in terms of pre/postconditions**
  
  e.g., room-booking activity:

  ```
  BookMeetingRoom (room, time, who)
  pre: not (exists w) Booked (room, time, w)
  post: Booked' (room, time, w)
  ```

  "Precondition" provides a predicate that must be true in the state prior to the initiation of the process

  "Postcondition" describes what will be true after the process execution

  Predicates and functions are unprimed/primed depending on whether they refer to the state before/after the activity respectively.

  Specialization of activities:
  
  e.g., To book a talk, we also need to book a projector

  ```
  BookTalk (room, time, who) IS-A BookMeetingRoom
  pre: not (exists r) AVRequested (r, time, overhead)
  post: AVRequested' (room, time, overhead)
  ```