More Dynamic Modeling

Review of getting started:
- Scenario making: gets us started thinking about events.
- Interface (high-level prototyping): helps us to think about order of things (happening in projects).
- Event trace: helps to know what object is doing what action.
- State Diagram creation tips.

Dynamic Model - State Diagram
- Graphical representation of a finite state machine.
- Changing states - transitioning on events.
- Events - external stimuli and internal messages
  - ex. button pushed; timer complete; tub full.
  - ex. "complete" event sent by state
- In each state, a set of predicates based on instance variables, is valid.

States Labeled with Conditions

Dynamic Models for E.S.
- Dynamic Model for user buttons would be simplistic; modeling might not be needed.
- Some environmental units might have behavior that should be modeled. (like an engine shifting through speeds)
- For embedded systems - might only need one significant behavior model (for controller.)
- Complex models will be decomposed into more detailed behavioral models.
- Concurrency could be present within a model.

How dynamic model relates to object model
- One state diagram for each class (with important behavior.)
- Each class has concurrent behavior.
- Aggregation in the Object Model usually implies concurrency in the Dynamic Model.
**Examples of Aggregation (5.17)**

- **Object model**
  
  ![Diagram illustrating object model for Car, Ignition, Transmission, Brake, and Accelerator.]

  Each class here will need a concurrent state diagram.

**How to model concurrency within an object**

![Diagram showing concurrency in Car, Transmission, and Brake.]  

**How to hide complexity**

- Not have a ‘flat’ state diagram
- Start abstract and then do subdiagrams.
  - use bull’s eye
- Take one abstract state and expand it with state generalization.

**Example of nesting (and other syntax as well)**

![Diagram illustrating nesting and various states and transitions.]  

**State Generalization**

![Diagram showing state transitions for Forward, Neutral, Push R, Push N, Reverse, First, Second, and Third states.]  

**Notation on Transitions and in States**

- **Do activity**
  - takes some time
  - associated with a state.
- **Guards**
  - conditions - boolean
  - \{ guard \}
- **Actions**
  - instantaneous
  - associated with an event.
  - action
**Checking for completeness and consistency**

- Formal specifications do this better!
  - The mathematical format can allow automation of these types of checks.
- Every state should have a way in and out.
  - unless starting point or ending point.
- Look for one object's Dynamic Model sending an event that doesn't have any receiving transition in another object's DM.

**Things to watch out for**

- Think about input from concurrent objects at unexpected times.
  - ex. If more than one ATM machine is trying to access the same account at the same time.
- User input when not planned. (OK to ignore, but make sure that is what is really wanted.)
- Take your scenarios and see if they work!
  - Walk through seeing that all the object's operations/messages has all the needed transitions.

---

**Producer-Consumer - Normal Scenario**

```
producer
get
ack
data
ack
```

**Producer-Consumer - State machine 1**

```
consumer
start
send(producer.get)
ack
setbuf
wait
send(producer.ack)
data
send (consumer. ack)
```

**Producer-Consumer - State machine 2**

```
consumer
start
send(producer.get)
ack
setbuf
wait
send(producer.ack)
data
send (consumer.ack)
```

---

**Basic Class Diagram**

```
consumer
data, ack, ack
producer
get, ack
```

**Producer-Consumer - State machine 1**

```
consumer
start
send(producer.get)
ack
setbuf
wait
send(producer.ack)
data
send (consumer. ack)
```

**Producer-Consumer - State machine 2**

```
consumer
start
send(producer.get)
ack
setbuf
wait
send(producer.ack)
data
send (consumer.ack)
```
Dynamic Model Timing and Exceptional Handling

Topics Covered:
- Dynamic Model
  - Synchronization schemes
  - Exception Handling
  - Timing including safety critical issues.

Synchronization
- In concurrent processing, the actions of the objects are rarely independent of each other.
- One may need to stop and wait for another process to ‘catch up’ or get to a certain state.
- Example: In a nuclear power plant, the model would need to reflect waiting for rods to be in place before generating power.

(Very Simple) Power Plant

Synchronization of States by status detection
Transition between B1 and B2 will not fire until object A has entered state A2.

Synchronization of States by a common event
Firing of the two transitions in the two models will happen at the same time.
Synchronization of States by common data

Transition from State B1 to State B2 will not fire until in State A2. (This assumes shared memory.)

Exception Handling

- Events such as resets and hardware interrupts must be handled.
- These are called Exceptions.
- Needed if user can exit a sequence of states at anytime.

Examples of exception handling

- Possible to modeling exiting all the substates of a superstate in UML.
  - Ex. Pushing the N (neutral button) in any of the forward states of a transmission.
- 3 ways to exit: normal completion, direct transition, and exception.

Timing Issues in Dynamic Model

- Sometimes the firing of a transition is time dependent, especially in embedded systems.
- Real-time systems might have transitions that are tied to a real-time clock.
- States might time-out after a certain length of time.
- Transitions might need to be stalled for a certain length of time.

Timing (Safety critical)

- Safety critical real-time solutions
  - example:
    - transition out of ‘boiler on’ state after being in this state for 1 hour, even if one expects a transition on when(temperature>=expected).
Delays in Dynamic Model

- Sometimes a transition should not be fired for a certain amount of time.
- This timing constraint can be modeled using timeout and an extra state
  - ex. 10 seconds since the exit from state A
  - This will delay the transition to State B for 10 seconds.

More Timing Issues in D. M.

- For a real-time system, the event might refer to a real-time clock
  - example: changing a traffic signal from day operation to night operation at 10 p.m.