

More Dynamic Modeling

10a-MoreDynamicModels 1

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.

10a-MoreDynamicModels 2

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.

10a-MoreDynamicModels 3

States Labeled with Conditions

Microwave Oven

10a-MoreDynamicModels 4

Dynamic Models for E.S.

- Dynamic Model for user buttons would be simplistic; modeling might not be needed.


```

      graph LR
      on[on] -- "button pushed" --> off[off]
      off -- "button pushed" --> on
      
```
- 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.

10a-MoreDynamicModels 5

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.

10a-MoreDynamicModels 6

Examples of Aggregation (5.17)

- Object model

```

classDiagram
    class Car
    class Ignition
    class Transmission
    class Brake
    class Accelerator
    Car o-- Ignition
    Car o-- Transmission
    Car o-- Brake
    Car o-- Accelerator
  
```

Each class here will need a concurrent state diagram

10a-MoreDynamicModels 7

How to model concurrency within an object

The diagram shows four concurrent state diagrams for the Car object:

- Ignition:** States: off, starting, on. Transitions: turn key to start (off to starting), [Transmission in Neutral] (starting to on), release key (on to starting), turn key off (starting to off).
- Transmission:** States: Neutral, Reverse, Forward (sub-diagram). Transitions: push R (Neutral to Reverse), push N (Neutral to Reverse), push F (Neutral to Forward), push N (Reverse to Neutral), upshift (Forward to second), downshift (Forward to first), upshift (second to third), downshift (third to second).
- Accelerator:** States: off, on. Transitions: depress accelerator (off to on), release accelerator (on to off).
- Brake:** States: off, on. Transitions: depress brake (off to on), release brake (on to off).

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.

10a-MoreDynamicModels 9

Example of nesting (and other syntax as well)

The diagram shows a state diagram for a vending machine with nested activities:

- idle** state transitions to **Do/add to balance(value)** on *coin in(value)*.
- Do/add to balance(value)** transitions to **Do/test item present; make change** on *[item empty]*.
- Do/test item present; make change** transitions to **Do/dispense change** on *[change<0]* and to **Do/move arm to correct row** on *[change=0]*.
- Do/dispense change** transitions to **idle** on *[change>0]*.
- Do/move arm to correct row** transitions to **Do/move are to correct column**.
- Do/move are to correct column** transitions to **Do/push item off shelf**.
- Do/push item off shelf** transitions to **idle**.
- idle** also transitions to **idle** on *cancel / refund coins*.

Example: lower-level state diagram for Dispense item activity

10a-MoreDynamicModels 10

State Generalization

The diagram illustrates state generalization for a transmission system. It shows a detailed state diagram for the 'Forward' gear with states 'First', 'Second', and 'Third' and transitions for 'upshift' and 'downshift'. This is generalized into a higher-level state diagram where 'Forward' is a single state with transitions for 'Push F' and 'Push N'.

10a-MoreDynamicModels 11

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

State1 event1 (attrs) [condition1] /action1

State2 do/ activity 2

You might need any or all of these for your project!

10a-MoreDynamicModels 12



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.

10a-MoreDynamicModels

13



Things to watch out for



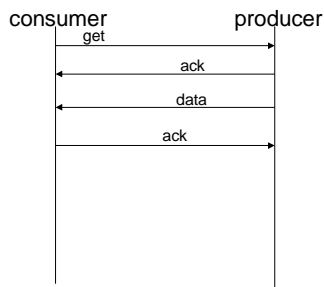
- Think about input from concurrent objects at unexpected times.
 - ex. If more than one ATM machine is trying to access the same account at the same time.
 - User input when not planned. (OK to ignore, but make sure that is what is really wanted.)
- Take your scenarios and see if they work!
 - Walk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

10a-MoreDynamicModels

14



Producer-Consumer - Normal Scenario

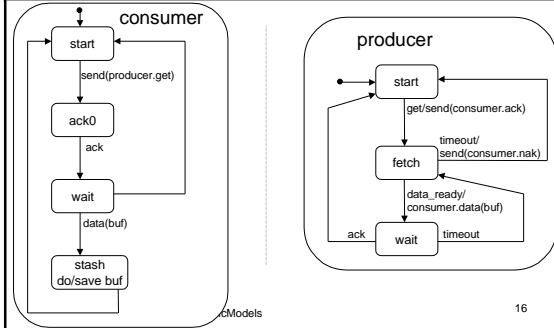


10a-MoreDynamicModels

15



Producer-Consumer State machine 1

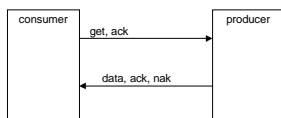


Models

16



Basic Class Diagram

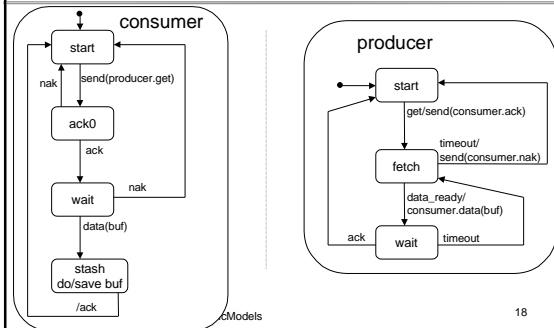


10a-MoreDynamicModels

17



Producer-Consumer State machine 2



Models

18

Dynamic Model Timing and Exceptional Handling



Topics Covered:

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

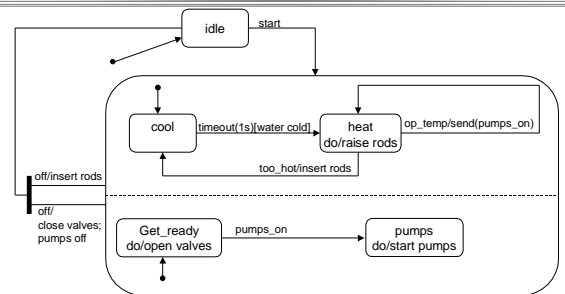


Synchronization

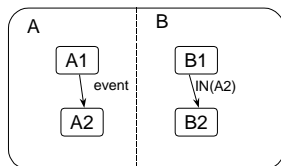
- In concurrent processing, the actions of the objects are rarely independent of each other.
- One may need to *stop and wait* for another process to 'catch up' or get to a certain state.
- Example: In a nuclear power plant, the model would need to reflect waiting for rods to be in place before generating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



(Very Simple) Power Plant



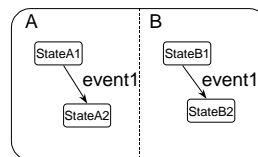
Synchronization of States by *status detection*



Transition between B1 and B2 will not fire until object A has entered state A2.



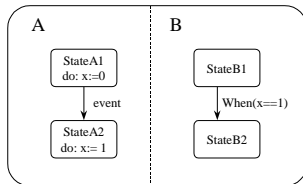
Synchronization of States by a *common event*



Firing of the two transitions in the two models will happen at the same time.



Synchronization of States by *common data*



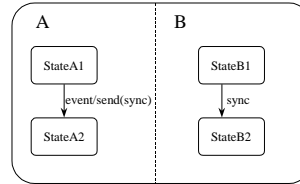
Transition from State B1 to State B2 will not fire until in State A2. (This assumes shared memory.)

10a-MoreDynamicModels

25



Synchronization of States by *Communication*



10a-MoreDynamicModels

26



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.

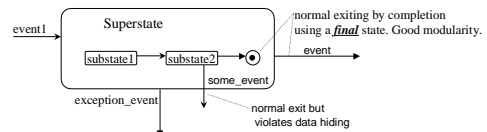
10a-MoreDynamicModels

27



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*.



10a-MoreDynamicModels

28



Timing Issues in Dynamic Model

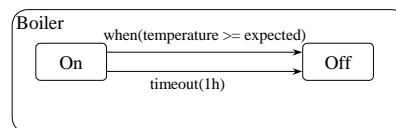
- Sometimes the firing of a transition is *time dependent*, especially in embedded systems.
- Real-time systems might have transitions that are tied to a *real-time clock*.
- States might *time-out* after a certain length of time.
- Transitions might need to be *stalled* for a certain length of time.

10a-MoreDynamicModels

29

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.



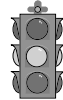
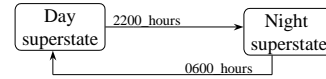
10a-MoreDynamicModels

31



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.



10a-MoreDynamicModels

32