**Embedded Systems Details**

**Object Model: Four main system objects or classes**

- **Controller object**
  - Might be made up of several controllers.
  - Is the brains of the system.
  - Takes input from the sensors and gives instructions to the actuators.

- **Sensor object**
  - Environmental objects that gives information to controller.
  - Can be passive (thermometer) or active (button).

**Object Model: Four main system objects (continued)**

- **Actuator object**
  - Environmental objects that are controlled by the controller.
  - Can be turned on or influenced by controller.
  - Examples: User indicator lights, motors, burners.

- **User Interface object**
  - A display for the user.
  - Can be made up of both sensors and actuators.
  - Example: machine control panel

**Step One: Develop a high-level object model**

- A dynamic model is a type of state machine.
- System can only be in one state at a time.
- Arrows: Transitions
  - From one state to another happen when events happen.
- Events are labeled on the transitions.
- Guards are conditions that keep a transition from happening, such as [is in neutral or park ]

**Step Two: Develop a system-level dynamic model**
Example: Automotive Door Control

- The system controls the windows and door locking functions.
- All doors have window roll up and down controls.
- Driver’s door has window lock feature.
- Driver and front passenger have door lock and unlock toggle.
- Fob unit for locking and unlocking doors, with driver notification (horn honk and lights flash.)
- Three concurrent systems identified.
Summary of development process

- The object model shows the real world objects grouped in classes with attributes and operations and associations.
- The dynamic model shows the control aspects with superstates refined into substates.

Embedded Systems Design

More Detail on Process

Review of Embedded Systems

- Software controller that is interacting with its hardware environment through sensors and actuators.
- Concurrency and real time issues.
- Safety critical nature of many of these systems.
- Increased demand for these systems to be designed well.
Object Model

- In UML, the object model is the starting place for the modeling process.
- The object model will include objects and their relationships.
- The Object Model will be the static, structural aspect of the system.

Identify Real World Objects

- Read over the problem description and find the nouns in the text.
- These nouns are candidates for objects in your object model.
- Discard unnecessary and incorrect classes.
- Object classes will include the controller (software unit that will be built), sensors, and actuators.

Data Dictionary: needs to be written

- A written paragraph describing each modeling entity.
- Needed so that names are non-ambiguous.

Class: Sensor

- Because of the common properties of all sensors, this can be a class of objects, called a superclass.
- Generalization - this superclass can be generalized into subclasses.
- Inheritance - each subclass will inherit the properties or features from the superclass.
- Examples: user interface (buttons etc.), thermometers, hardware sensors.

Class: Actuator

- Similarly, the actuators will probably become a superclass.
- Generalization - the various actuators can be generalized into subclasses.
- Inheritance - each actuator subclass will inherit properties or features from the superclass.
- Examples: LEDs, motor controls, etc.

The Controller

- At an abstract level, this would be only one object in most embedded systems.
- This object would be refined at lower levels of the modeling process into subsystems or sub-objects.
- Aggregation could be used to show the parts of the controller.
Model itself

- Graphically a class is shown as a box with the name on top.
- Attributes (middle third) and operations (bottom third) added eventually.
- Attributes and operations are not needed for high-level object model.

Find the Associations

- Interaction between objects must be shown by associations or lines draw with labels.
  - ex: line between user button and associated LED.
- Many times these associations will be a physical connection between objects in an embedded system.
- Multiplicity must be shown eventually.

Example

![Diagram of Example]

Conclusion about Object Model:

- Not very complex at first.
- More details will come as designer proceeds from abstraction to more and more concreteness.
  - controller will be divided into more objects
  - attributes and operations are identified and included.
- Starting place for OO Modeling. Sets the stage.

Next step: Dynamic Model

- The dynamic model shows the control aspect of the system.
- Because embedded systems are mainly controllers, the dynamic model is the 'key' model for embedded systems.
- This model can show the timing aspects.
- Shows sequence of operations in response to external stimuli.

Getting started on a Dynamic Model

- Helpful to make a scenario:
  - sequence of events that happens in one execution of a system.
  - Example: insert coins, make selection, pop dispensed.
- Interface (high-level prototyping)
  - a rough draft of user interface will help thinking about the events in an embedded system.
continue getting started….

- Next make an event trace.
  - each object is a vertical line.
  - events as horizontal arrow.
  - time goes from top to bottom.
- Use previously discussed ‘creation tips.’

Example of an Event Trace

<table>
<thead>
<tr>
<th>User</th>
<th>ATM</th>
<th>Consortium</th>
<th>Bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert card</td>
<td>request password</td>
<td>verify card with bank</td>
<td>bank account OK</td>
</tr>
<tr>
<td></td>
<td>request password</td>
<td>account OK</td>
<td>bank account OK</td>
</tr>
<tr>
<td>request kind</td>
<td>enter kind</td>
<td>process transaction</td>
<td>bank transaction succeed</td>
</tr>
<tr>
<td>request amount</td>
<td>enter amount</td>
<td>process transaction</td>
<td>bank transaction succeed</td>
</tr>
</tbody>
</table>

Event trace for ATM scenario
Example from Figure 8.18 of Rumback