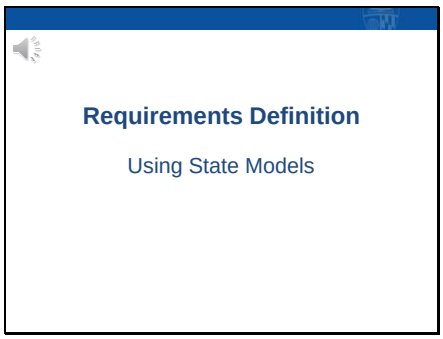
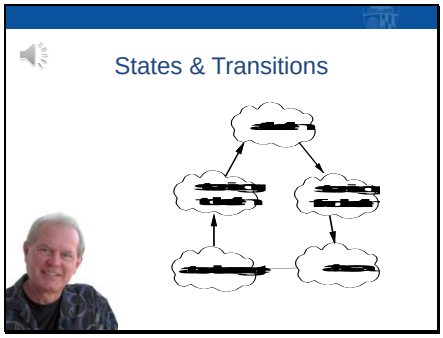
1



In this lecture we'll discuss how to use state models to document how state changes occur in a product or product component.

2



Some types of applications are often state-full…and by state-full I mean that an application goes through a series of state changes over the course of processing time. Simply put…a state is a set of conditions at some point in time…and those conditions can change over the course of time based upon things that occur. Let's take an example.

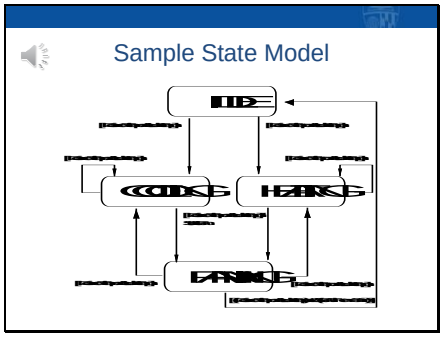
Suppose we were defining requirements for a system that would track the processing of ships at a commercial port. A ship would have state-full behavior, and it would be important for us to identify, document, and model that state-fullness for that type of application. For example, a ship could be at sea…meaning it hasn't arrived at the port yet. This could be important to know, since it could be used to estimate a ship's arrival time. Once a ship arrives at a port, it typically drops anchor near the official port entry point in an area known as an anchorage. It waits at the anchorage until it is given a berth assignment and a time it can leave the anchorage. When it departs the anchorage, it takes some time to reach its berth…the ship is said to be steaming to berth. While it's at the berth it is being processed…meaning that cargo is being loaded and/or unloaded. It then gets a departure time assigned to it, and it steams from the berth, past the port entry point, and returns to its at-sea state. By the way, this is a real example, but I've simplified the states for discussion purposes.

The states a ship is in, and the changes between those

states, are illustrated in this little diagram. The clouds represent the states a ship can be in, and the arrows indicate the state changes…or state transitions…associated with a ship. The states and transitions provide very important information. The transitions are important because they define the allowable state changes. If a ship is currently in one state, its next state is very specific…for example, if a ship is at anchorage its next state must be steaming to berth…it couldn't be steaming from berth. The states are important because certain processing functions are dependent upon specific states. For example, a ship must be at berth in order for cranes to be assigned to unload its cargo. Cranes can't be assigned if a ship is at sea or steaming from berth.

A way of documenting states and transitions is to use something called a state model…so let's see how that's done.

3



Here's a state model for a microprocessor-based HVAC system in a smart home. When it's turned on, the system will automatically control the heading and air conditioning.

The ovals represent states the system can be in, and the arrows represent state changes…or transitions. Now…this model represents the HVAC system when it's in its on state. That is, the system is turned on, and the thermostat is monitoring the temperature in a particular area. If the system is turned off…it does nothing. Now I could have shown an off state but I didn't for simplicity.

Events can happen that can cause state transitions to take place. Events can be external to the system…like someone pressing an on/off switch…or an event can be internal to the system. In this example, most events will be internal, and will have to do with comparisons of a desired temperature setting and the actual temperature sensed by the thermostat.

Let me walk you through the state model. When the

system is turned on, the IDLE state is the initial state of the system. In the IDLE state, the thermostat is monitoring the temperature. If it gets a signal that it is too warm in the room, it will turn on the air conditioning. That is reflected in this model by making a transition from the IDLE to the cooling state. It remains in the cooling state until the room cools down to the desired temperature. This is reflected in the arrow on the left that begins and ends in the COOLING state…this is called a reflexsive or self transition. When the room reaches the desired temperature, a fan is turned on to blow the cool air around. This is reflected by the FANNING state. If the desired temperature is maintained, it will remain in the FANNING state for a certain period of time, after which the fan will be turned off (to save energy) and the system will be returned to the IDLE state.

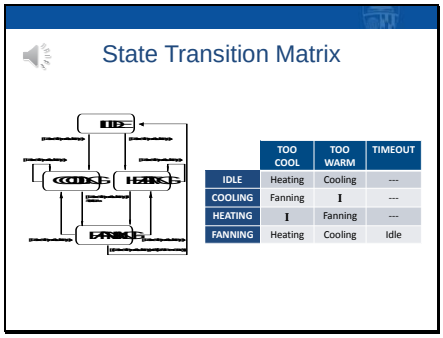
When in the IDLE state, if the room becomes too cool, the heater may be turned on, and the specified set of state changes illustrated on the right side of the diagram will occur.

I didn't go through all the possibilities in this walk through, but hopefully you get the idea. Now, this state model is a lot more complicated than in the earlier ship example...but it's very realistic.

I'm sure you noticed the expressions in the square brackets. They're called guard conditions…and they've been used here to provide more specifics about the conditions under which state transitions can take place. Guard conditions are boolean expressions. If a guard condition is true, a state change can take place. If it is false, the state change cannot take place. I'll walk through one of the guard conditions for the FANNING state. It states that a transition to the IDLE state will take place if the desired temperature equals the actual temperature and the cumulative time the fan has been on is three minutes or longer. In other words, the fan will blow the air around as long as it is the desired temperature, but only for three minutes, before the IDLE state resumes.

By the way, the notation I'm using here is defined by the Unified Modeling Language, or UML. You'll learn more about the UML as the course progresses. Also, there is a lot more involved in state modelling, but I think our discussion is more than sufficient for purposes of this course.

4



	TOO COOL	TOO WARM	TIMEOUT
IDLE	Heating	Cooling	---
COOLING	Fanning	I	---
HEATING	I	Fanning	---
FANNING	Heating	Cooling	Idle

Another type of model that is often used in conjunction with a state model, or in lieu of a state model, is a state transition matrix.

Here, I've illustrated a state transition matrix for the HVAC system example. In this matrix there is a row for each state and a column for each event. The too cool even occurs when the desired temperature is greater than the actual temperature, the too warm event occurs when the desired temperature is less than the actual temperature, and the timeout event occurs if the bottom-most guard condition of the FANNING state is true.

I'll walk you through one or two rows of the table and you can see how it maps back to the state model. Let's suppose the system is in the IDLE state and a TOO WARM event occurs. The table tells us that the system makes a transition to the COOLING state. If the system is in the COOLING state and a TOO WARM event occurs, it ignores the event…I used a big I to indicate that…and ignoring the event corresponds to the reflexive transition on the state model. If the system is in the COOLING state and a TIMEOUT event occurs…well that can't happen, since a timeout can only occur when the system is in the FANNING state…so I use a blank cell to indicate that it can't happen.

By the way, I used events for the columns because I wanted to introduce the event concept. I could have

used the guard conditions there instead.

So…in summary…I hope you can see how state modeling can be a very useful activity for certain types of systems. In practice, some organizations defer state modeling to the design phase, but technically it can be considered to be a requirements activity. When we use an object-oriented approach to system development, state modeling is a common requirements analysis activity.