Dynamic Modelling and
Process Re-engineering using UML

Part 2 – State Machine diagrams

Written by: Robin Beaumont e-mail: robin@organplayers.co.uk

Date last updated: Wednesday, 07 September 2011
Version: 1

How this document should be used:
This document has been designed to be suitable for web-based and face-to-face teaching. The text has been made to be as interactive as possible with exercises, Multiple Choice Questions (MCQs) and web-based exercises. If you are using this document as part of a web-based course you are urged to use the online discussion board to discuss the issues raised in this document and share your solutions with other students.

This document is aimed for two types of people:
• Those who wish to become involved in database development or process re-engineering but are not interested in the nuts and bolts of programming. Such people are commonly called domain experts and act as bridges between a professional group (e.g. medics, solicitors etc) to which they belong and IT experts.
• As an introduction for those just beginning professional computer science courses

Acknowledgements
I would like to thanks all the students that have used this material from the early 1990's up until the present who have provided invaluable, references, comments and suggestions.

I hope you enjoy working through this document.

Robin Beaumont
Contents

1. Learning Outcomes ........................................................................................................................................3

2. Preparation ......................................................................................................................................................3

3. Introduction ....................................................................................................................................................4

4. States in general ..............................................................................................................................................5

5. States in UML ..................................................................................................................................................6

6. Transitions .......................................................................................................................................................7

7. Triggers (Events) ..............................................................................................................................................7
   7.1 Guard constraints ........................................................................................................................................8
   7.2 Initial and Terminal States ........................................................................................................................9
   7.3 Nesting ........................................................................................................................................................9
   7.4 Timed Triggers ..........................................................................................................................................8

8. Obtaining an Initial State Diagram from a Sequence Diagram .....................................................................10

9. Multiple Choice Questions (MCQs) ...........................................................................................................11

10. Summary ......................................................................................................................................................12
1. Learning Outcomes

This document aims to provide you with the following skills and knowledge. After you have completed it you should come back to these points, ticking off those with which you feel happy.

<table>
<thead>
<tr>
<th>Learning outcome</th>
<th>Tick box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Be aware that there are two types of State diagram in UML 2.4</td>
<td></td>
</tr>
<tr>
<td>Be able to describe the diagramming complexities of (behavioural) State machine diagrams</td>
<td></td>
</tr>
<tr>
<td>Be aware of the concept of ‘configuration’ within dynamic modelling</td>
<td></td>
</tr>
<tr>
<td>Be able to describe the relationship between Triggers (events) and states</td>
<td></td>
</tr>
<tr>
<td>Be able to describe what a state diagram (Higraph) is</td>
<td></td>
</tr>
<tr>
<td>Be aware that it is possible to develop state diagrams from sequence diagrams</td>
<td></td>
</tr>
<tr>
<td>Be able to use condition statements in state diagrams</td>
<td></td>
</tr>
<tr>
<td>Be aware of the incremental nature of dynamic model development</td>
<td></td>
</tr>
<tr>
<td>Be able to explain the ‘initial’, ‘terminal’ and exception symbols</td>
<td></td>
</tr>
<tr>
<td>Be able to explain decomposition/nesting</td>
<td></td>
</tr>
<tr>
<td>Be aware of the importance of dynamic modelling in the health sector</td>
<td></td>
</tr>
<tr>
<td>Describe the process involved in developing a dynamic model</td>
<td></td>
</tr>
</tbody>
</table>

2. Preparation

Having worked through the previous UML chapters discussing Classes, Instances, associations, use cases and sequence diagrams (http://www.robin-beaumont.co.uk/virtualclassroom/contents.html) you will understand a wide range of UML concepts (Unified Modelling Language). While the previous chapters focused on the ‘what’, and the use case/sequence diagram chapter on the ‘how’ aspect of modelling all the chapters have tended to concentrate on the interaction between various classes/instances however it is now time to consider in detail the inner lifecycle of a particular class for any particular instance it may spawn. This aspect is modelled by using State machine diagrams in UML. Technically there are two types of State diagrams in UML 2.4 behavioural state machine and Protocol state machine diagrams, we will only consider the behavioural state machine each of which is owned by a context which for us means a class. For example in the diagram below we can see that the Nurse class owns a Nurse state diagram.

For the rest of this chapter we will ignore Protocol state machine diagrams and focus on behavioural state machine diagrams.
3. Introduction

One of the diagrams UML uses to model the dynamic nature of an individual class is the Behavioural State [Machine] Diagram, often referred to simply as a state diagram. State diagrams have been used in a number of forms for well over four decades.

State diagrams, State machine diagrams and Harel's Higraphs all refer to the same basic diagram type. However UML has added an array of additional symbols to the diagrams to provide a greater degree of expression. But let's begin with basics in this section.

State diagrams show visually the relationships between transitions, states and events for a single class. A simple example of a state diagram is given opposite.

The first and possibly most important thing to note is the title. The state diagram relates to one particular class, in this example that of the GP class which also relates to the Consultation SD described earlier. Comparing the above state diagram with that of the sequence diagram produced in the previous chapter we note that there are similarities; the messages which possess the arrows on the sequence diagram that point towards the GP instance appear to roughly equate to the states in the above diagram. We can also surmise that any messages which neither originate nor terminate at the GP instance such as go_to_nurse (in the nurse consultation interaction use) do not appear in the above diagram, but much more of this later.

The rounded boxes represent states, and the lines represent transitions some of which appear to have names on them which equate to triggers. Basically a quick first explanation of the three concepts within the UML state diagram might be:

State - a particular configuration of an instance over a time period which may be terminated or initiated by an event.

Transition – the change from one state to another.

Trigger (Event) - something that happens at a point in time and can cause an activity/transition. A trigger was called an event prior to uml2 (superstructure specification 2.4 page 588). Now they are different but closely related as a trigger specifies an event that then may cause some type of behaviour.

As usual definitions mean little without us looking in depth at each of them, along with examples, which we will now do.

---

### Important Differences Between Sequence and State Diagrams:

<table>
<thead>
<tr>
<th>Sequence Diagrams</th>
<th>State Diagrams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examine communication across objects</td>
<td>Examine all possible states for one particular class</td>
</tr>
<tr>
<td>At the instance level</td>
<td>At the class level</td>
</tr>
</tbody>
</table>
4. States in general

A state is usually a result of a particular configuration of a class instance and lasts for some duration of time. Considering this sentence, what exactly is meant by 'duration of time' and a 'configuration'?

The duration of time is purely circumstantial given the particular concerns of the modeller; in other words, it is contextual. For example, a modeller interested in the development of large cities is not interested in the minute by minute changes that take place, such as individual traffic light changes. In contrast a modeller concerned with traffic control systems would be precisely interested in that aspect and not interested in long term changes. Computer scientists tend to use the word granularity to describe this aspect.

So what is a configuration?

This rather grand sounding word is for something that is quite simple. It just means the set of values that the data items for a particular instance have at a particular time. Let's consider some examples, we can say that a particular person (instance of class person) is in a particular state at any one time where the state is defined by a particular set of values the data items possess say age and developmental stage in the diagram below, so we have an eating child, idle adult and shopping granny states in the diagram. Alternatively considering a simpler situation, for example the hospital theatre class instance below, we can define the state by the value taken by a single data item that being the status data item with the value taking at least 'usable' or 'being_decorated'.

States can be nested for example the state shopping might include the (sub)states eating and not eating.

Examples of some Configurations

Some Configurations for a person:

Person
<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>age</th>
<th>Developmental stage</th>
<th>Hair colour</th>
<th>Hair state</th>
<th>Doing shopping</th>
<th>eating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Child</td>
<td>brown</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

PersonA:Person
| ID=001 | Name=John | Age=2 | Developmental stage=Child | hair colour=brown | washing hair=no | shopping=no | eating=yes |

PersonB:Person
| ID=002 | Name=mary | Age=44 | Developmental stage=adult | hair colour=brown | washing hair=no | shopping=no | eating=no |

PersonC:Person
| ID=003 | Name=Edith | Age=67 | Developmental stage=retired | hair colour=grey | washing hair=no | shopping=yes | eating=no |

Some Configurations for a HOSPITAL theatre:

Theatre
<table>
<thead>
<tr>
<th>Status</th>
<th>Full</th>
<th>Accept_patient</th>
<th>Reject_HIV_patient</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP:Plastics:Theatre</td>
<td>Status=usable</td>
<td>Full=no</td>
<td>Accept_patient=yes</td>
</tr>
<tr>
<td>B:General:Theatre</td>
<td>Status=usable</td>
<td>Full=yes</td>
<td>Accept_patient=no</td>
</tr>
<tr>
<td>C:Trauma:Theatre</td>
<td>Status=being_decorated</td>
<td>Full=no</td>
<td>Accept_patient=no</td>
</tr>
</tbody>
</table>

States can be considered to fall into two main types: idle or active (i.e. performing some type of activity).

States also relate to various other concepts in UML such as activities (behaviours) and actions, within the UML each of these has very specific but rather confusing meanings and various versions of UML have changed the meanings of them over the time. I will try to steer the content of this chapter away from the more contentious murky waters.
Exercise 1. Identifying States

Time: 20 minutes maximum

1. Consider a role within your organisation and draw up a list of possible states.

2. States are often modelled as a data item in an instance which can take a number of values. For example, considering a doctor, an instances might be within any of the following states (student, intern1, intern2, spr, junior consultant, senior consultant, retired), resulting from various configurations. Can an instance be in more than one state at any one time? If this is so, how could it be modelled considering the above information about the doctor?

5. States in UML

Within UML State machine diagrams states are usually named using active (continuous) verbs (http://www.englishpage.com/verbpage/activepassive.html) meaning that they usually end in 'ing', for example looking at the state diagram example at the beginning of this chapter we see names like welcoming, sending, reassuring etc. which is typical, however this is not always the case, consider a state diagram for a light, which is in one of two states on or off!

As with most things in UML the state symbol can have additional semantic enhancements, besides the transition name (which is optional!). The following description provides additional information and has been taken and adapted from the 2.4 superstructure specification.

The compartments of a state are, where the second two appear together:
- name compartment
- internal activities compartment
- internal transitions compartment

**Internal activities compartment**

This compartment holds a list of internal actions or state (do) activities (behaviours) that are performed while in the state.

- **entry** — This label identifies a behaviour, which is performed upon entry to the state (entry behaviour).
- **exit** — This label identifies a behaviour, that is performed upon exit from the state (exit behaviour).
- **do** — This label identifies an ongoing behaviour (“do activity”) that is performed while in the state or until the computation specified by the expression is completed (the latter may result in a completion event being generated).

**Internal transition compartment**

This compartment contains a list of internal transitions, where each item has the form as described for a Trigger. A trigger defines types of events that can initiate a transition between states. Think of triggers as types of events (i.e. something that causes something to happen), of which there are more than ten types defined in the uml. One particular trigger type called the callEvent is interesting as it allows the calling up of an operation, so here you can indicate if a particular operation is active during this state. In the example above I have indicated that the examination() operation is called and becomes active while in the examining state.

You will notice that I also have the operation on the transition line going into the state – this is because I felt that really it should belong to the transition line as by invoking the operation the instance state would change to examining, they wouldn’t be in the examining state and then invoke it?

For simplicity I would ignore the entry/do/exit behaviours, pre uml 2.4. you could specify methods linked to them however it appears now no longer possible as they link to behaviours which can then link to operations – just too complex.

It is important to realise that the trigger/event name is not shown on the diagram it is the associated operation or activity names that are displayed
6. Transitions

A transition is a response to a trigger/event which takes the state diagram from one state to another.

I would like to introduce here a particular method that modellers use to formally specify their language called Extended Backus–Naur Form (EBNF) this is used throughout the computing community to communicate and also in the UML superstructure, for example it is used to define a trigger and a transition p. 592:

So considering the above diagram, a transition in UML 2.4 is defined as an expression that has three optional expressions, zero or more triggers, a guard and a behaviour expression. Similarly a trigger is defined to be either a call-event, signal event, any-receive-event, time-event or a change-event. But enough of the theory (which you can now forget) and lets see what this means in practice.

A behaviour expression is rather a loose concept in this context it often is used to indicate a updated value for a particular attribute etc for example we might have something like patient_sexual_status=active.

7. Triggers (Events)

While the above paragraph provides a formal definition of what a Trigger is in UML it is easier to get a handle on the concept by considering several examples.

The example opposite is possibly the simplest one you might come across, a named trigger nudges the movement from state a to state b and also state c.

If the particular trigger had been a type that causes an operation to fire (as is the case with a Call event type trigger you would expect a set of parentheses after the name.

In uml since version 2 you do not define events as such just the specific trigger (or rather the subtype of trigger as defined above) that is associated with the event.
Exercise 2. Drawing State Diagrams

Time: 30 minutes maximum

1. From the list of roles/states you developed in the last exercise, think of possible triggers that cause the transitions. Create a simple state diagram like the one on this page from your findings.

2. Drawing inferences from the diagram opposite, what would the life of a light switch look like, assuming it will always exist?

3. How would you show a trigger that did not change a state? [Hint: think about recursion.]

7.1 Timed Triggers

You can also specify timed events using such words as **when** or **after**

- after (5 seconds)
- after (10 seconds since exit from state A)
- when (date = Jan. 1, 2000)."

You can also specify values to use such as timed_out etc.

We have now covered some of the more common diagramming options of the state diagram. Let’s move on to thinking about how you might produce one from a Sequence Diagram.

7.2 Guards/constraints

These are exactly what the name implies. A condition statement acts as a guard, allowing the transition to occur only when the condition statement is true. A condition can therefore act to prevent moving to the next state or allowing movement to the next state (when the condition is true). It is shown on the state chart diagram as a junction symbol with a statement enclosed in square brackets, as in the example opposite.

What do the two additions to the diagram mean?

- 'move to the reassuring state, if patient is weepy (condition)'.
- 'move to the send to nurse state if the patient is not weepy (condition)'.

You should notice that in the above state diagram there is no other text on the transition lines. This indicates that there is automatic movement from one state to another when the activity has completed. This is called **triggerless transition**. This is in contrast to the state diagram at the beginning of this state diagram section (see page Error! Bookmark not defined.) where an event name occurs on most of the transition lines indicating that the transition is fired off by a trigger. For example the event Patient_arrives triggers the transition to the examining state etc.
8. Initial and Terminal States

The above state diagram also shows two new symbols: those that indicate the 'start' and 'end' of a particular trigger state sequence.

You should realise that the first state is the one that the arrow points to i.e. Alive. The black bubble does not represent a state itself.

Therefore the first state diagram is wrong if you consider dead to be a state!

Key:
- = start
- = end
[ . . ] = condition

The final state can also be reached from a number of transitions

9. Nesting

A state diagram can contain other state diagrams (called children). In the example provided in this document regarding the GP, the process 'examining' in the diagram might be blown up into another whole state diagram providing further details. Clearly the inputs and outputs to the child state diagram must reflect those of the parent state. Children state diagrams can be attached to either states or events of the parent diagram.

The concept of nesting is called, more correctly, decomposition and is an essential aspect of systems modelling. State diagrams can get very messy if you show all the states so it is lucky that UML provides a method of indicating that substates exit but are in another state diagram.

The above state diagram for Nurse Consulting shows three parallel states reflecting the Sequence Diagram we created earlier. The circle with a cross indicates a **exit point pseudostate** this is used to indicate that it is possible to exit the state if an abnormal condition occurs.
10. Obtaining an Initial State Diagram from a Sequence Diagram

The task of developing a state diagram from a Sequence Diagram is not a trivial task, and both editions of Rumbaugh’s book unusually does little to help. In contrast, Pender 2002 (pages 200, 238-244) has by far the best explanation I have come across so far. The following is largely my personal experience as well as reading between the lines. My technique is as follows:

1. Consolidate the Sequence Diagrams. The one below only has asynchronous signal messages, the technique is simpler for call messages.

2. Taking an individual instance in the Sequence Diagram (i.e. doctor in diagram below):

   • Consider those asynchronous signal messages for which it is a target (arrows towards it), and consider these provisionally as triggers for the class (instance).
   • Consider those messages for which it is the originator (arrow away from it), and consider these provisionally as states for the instance.

Looking back once again at the Sequence Diagram for the patient, GP and nurse it can now be considered to be just one path through several state diagrams only one of which we have considered.

---

### candidate states for the Doctor class

- **requesting patient**
- **welcoming patient**
- **requesting information**
- **reassuring**
- **Sending to nurse**
- **Providing results**

---

### Interaction: consultation single branching

- **AbdulHussein : patient**
- **AmySmith : doctor**

1: **Request_patient**

2: **welcomes**

3: **acknowledges**

4: **requests information**

5: **provides information**

6: **cuddle and reassure**

7: **reassure**

8: **give paper handout**

9: **send_to_nurse**

10: **request results**

---

**Red arrows = candidate triggers for the doctor class**
Exercise 3. Converting Sequence diagrams to State Diagrams

1. Using one of the Sequence Diagrams you developed earlier, take one of the instances in it and create a state diagram for it.

Remember the appropriate rules given above concerning which messages on the sequence diagram should be considered to be states or states within the state diagram. Notice that you may want to enhance the state diagram once you start to looking in more detail. This is quite normal and to be expected.

Exercise 4. Suggestions for more State Diagrams

1. Considering the DopeHead scenario draw a state diagram for one of the classes you have identified. Details of how to obtain the modelling scenarios document is given in the “Before You Start” section at the beginning of this chapter.

2. Considering your role list the possible states you might be in and what triggers various transitions.

11. Multiple Choice Questions (MCQs)

I have included a number of multiple choice questions below to help you revise the material in this document. I would advise you to mark beside each one where you found the answer in the document.

1. States can be considered to fall into the following main categories:
   
   a. Idle/active
   b. Isolated/interactive
   c. preparing/active
   d. True/false/unknown
   e. Alive/dead

2. Which of the following is the most appropriate definition of a state diagram (Harel's Higraph)?

   a. State diagrams show visually the relationships between triggers (events) and states for several classes.
   b. State diagrams show visually the relationships between triggers (events) and states for a single class.
   c. State diagrams show visually the triggers (events) for a single class.
   d. State diagrams show visually the states for a single class.
   e. State diagrams show visually the relationships between triggers (events) for several objects.

3. The lines in state diagrams (Harel's Higraphs) represent:

   a. Signals
   b. Alarms
   c. States
   d. Transitions
   e. Initial conditions

4. When developing dynamic models, what is the usual sequence of activities?

   a. Sequence diagram -> state diagrams
   b. State diagram -> several Sequence Diagrams
   c. Sequence diagrams -> one state diagram
   d. State diagrams -> one Sequence Diagram
   e. State and Sequence Diagrams developed independently then finally consolidated
5. If you see the expression "[if weepy]" along a line in a state diagram, it would represent the following:
   a. Textual description (unnecessary)
   b. Message
   c. guard/Condition statement
   d. Event
   e. Activity

6. What is the correct term for nesting?
   a. Inclusion
   b. Decomposition
   c. Induction
   d. Fragmentation
   e. Deduction

7. Which of the following statements is correct:
   a. Decomposition is an important aspect of systems modelling.
   b. Decomposition is a minor aspect of systems modelling.
   c. Decomposition requires extra diagramming tools to detail it.
   d. Decomposition is not a conceptually easy topic.
   e. Decomposition is an optional component to modelling.

12. **Summary**

This chapter has focused on the inner life of a class, and brings us to the end of this introduction to UML.

Please see the previous chapter for a list of references and web links.