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

I hope you enjoy working through this document.

Robin Beaumont
Contents

1. Before You Start.................................................................................................................................. 4
   1.1 Required Resources .................................................................................................................. 4

2. Learning Outcomes ........................................................................................................................... 5

3. Introduction ....................................................................................................................................... 6
   3.1 Where Does the Dynamic Aspect Fit in the Modelling Process? ............................................. 7

4. Use Case Diagram ............................................................................................................................. 8

5. What is the Dynamic Aspect? ........................................................................................................... 12
   5.1 Messages/Events ..................................................................................................................... 14

6. Sequence Diagrams .......................................................................................................................... 15
   6.1 Frames ..................................................................................................................................... 16
   6.2 Lifelines .................................................................................................................................... 16
   6.3 The Message........................................................................................................................... 17
      6.3.1 Message Types .................................................................................................................. 17
      6.3.2 Slanted Arrows ................................................................................................................ 18
   6.4 Showing More Detail – Interaction Occurrence ....................................................................... 19
   6.5 Control ..................................................................................................................................... 19
      6.5.1 Iteration ............................................................................................................................ 19
      6.5.2 Single Branching ............................................................................................................... 20
      6.5.3 Multiple Branching .......................................................................................................... 21
      6.5.4 Concurrent Messages ...................................................................................................... 22
   6.6 Nesting ....................................................................................................................................... 22
   6.7 Incremental Development ......................................................................................................... 22
   6.8 Exercises - Sequence Diagrams ............................................................................................... 23

7. Business Process Re-engineering (BPR) .......................................................................................... 29
   7.1 Relationship Between Sequence Diagrams, Use Cases and Class Diagrams ......................... 30
      7.1.1 Originator Message Naming Style .................................................................................... 30
      7.1.2 Target Style Message Naming .......................................................................................... 31
      7.1.3 Use Cases ........................................................................................................................ 31

8. State [Machine] Diagrams (Harel’s Higraphs) .................................................................................. 33
   8.1 States ......................................................................................................................................... 34
   8.2 Relationship between Transitions and States .......................................................................... 35
   8.3 Condition Statements ............................................................................................................... 36
   8.4 Initial and Terminal States ....................................................................................................... 36
   8.5 Nesting ...................................................................................................................................... 37
   8.6 Other State Compartment Options ......................................................................................... 37
   8.7 Timed Triggers .......................................................................................................................... 38

9. Obtaining an Initial State Diagram from a Sequence Diagram .................................................. 39

10. Importance of Dynamic Modelling ............................................................................................... 40

11. Multiple Choice Questions (MCQs) .............................................................................................. 41

12. Summary .......................................................................................................................................... 45
Acknowledgment

The various UML diagrams in this document were drawn using three case tools, The student edition of System Architect, VP-UML (community edition) and MagicDraw (academic personal edition)
1. Before You Start

This document assumes that you have the following prerequisite knowledge and skills:

<table>
<thead>
<tr>
<th>Skills:</th>
<th>Where can I gain the skills and knowledge?</th>
</tr>
</thead>
<tbody>
<tr>
<td>That you have used the following features of a Database Management System (DBMS) such as Access or Paradox to:</td>
<td></td>
</tr>
<tr>
<td>• Create tables</td>
<td><a href="http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm">http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm</a></td>
</tr>
<tr>
<td>• Create relationships (and therefore know about the relationship window)</td>
<td></td>
</tr>
<tr>
<td>• Create simple queries in the query design window</td>
<td></td>
</tr>
<tr>
<td>• Create a simple form</td>
<td></td>
</tr>
<tr>
<td>That you have used a case tool such as DeZign or System Architect to create ERDs and UML Class diagrams</td>
<td><a href="http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm">http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm</a></td>
</tr>
<tr>
<td>Knowledge:</td>
<td></td>
</tr>
<tr>
<td>1. Concerned with databases:</td>
<td></td>
</tr>
<tr>
<td>• Tables, indexes and fields</td>
<td><a href="http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm">http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm</a></td>
</tr>
<tr>
<td>• Relationships (and the links between modelling and database implementation)</td>
<td></td>
</tr>
<tr>
<td>• Forms</td>
<td></td>
</tr>
<tr>
<td>• Queries</td>
<td></td>
</tr>
<tr>
<td>2. Concerned with modelling and object oriented modelling:</td>
<td></td>
</tr>
<tr>
<td>• Entity types and entity instances</td>
<td><a href="http://www.robin-beaumont.co.uk/rbeaumont/chap11/s2/omt1.pdf">http://www.robin-beaumont.co.uk/rbeaumont/chap11/s2/omt1.pdf</a></td>
</tr>
<tr>
<td>• Class and object models</td>
<td></td>
</tr>
<tr>
<td>• Associations including aggregation and generalisation</td>
<td></td>
</tr>
<tr>
<td>• Have an awareness but not in-depth knowledge of encapsulation and polymorphism</td>
<td></td>
</tr>
</tbody>
</table>

1.1 Required Resources

You need the following resources to work through this document:

- Many of the concepts introduced in this document are difficult to grasp at first and are helped by experimenting with a case tool in addition to carrying out the exercises with pen and paper. One such tool is Magicdraw. The document “An Introduction to ERDs” provides details of other websites where you download alternative software.


- If you are feeling particularly masochistic you can download the latest UML standards documents at: http://www.omg.org and follow the links.
2. Learning Outcomes

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

<table>
<thead>
<tr>
<th>Learning outcome</th>
<th>Tick box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explain what the 'dynamic aspect' of an object is</td>
<td></td>
</tr>
<tr>
<td>Know how the dynamic aspect of an object is modelled</td>
<td></td>
</tr>
<tr>
<td>Be able to explain the concept of events and states as used in object oriented (OO) modelling</td>
<td></td>
</tr>
<tr>
<td>Understand the components of a Use Case diagram</td>
<td></td>
</tr>
<tr>
<td>Be able to develop Use Case diagrams</td>
<td></td>
</tr>
<tr>
<td>Understand the components of a Sequence Diagram</td>
<td></td>
</tr>
<tr>
<td>Be able to develop Sequence Diagrams</td>
<td></td>
</tr>
<tr>
<td>Be able to draw Sequence Diagrams to carry out BPR</td>
<td></td>
</tr>
<tr>
<td>Be aware of the diagramming complexities of Sequence Diagrams</td>
<td></td>
</tr>
<tr>
<td>Be able to explain what business process re-engineering (BPR) is</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 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' and 'terminal' 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>
3. Introduction

Having worked through the “Introduction to Classes and Objects” document (http://www.robin-beaumont.co.uk/virtualclassroom/chap11/s2/omt1.pdf) you will understand a wide range of concepts concerned with class / object oriented modelling. That document focused on the what aspect of modelling, basically the information you would consider collecting to create classes and their relationships. In this document we will move to the how aspect of modelling, that is the life cycle of these classes and how they actually interact with one another over time. We will do this by considering three types of UML diagram:

- Use Case
- Sequence
- State [machine]

Defining what happens over time is not an easy subject! Rumbaugh 1991 begins his chapter on dynamic modelling with the warning ‘Temporal [dynamic] relationships are difficult to understand’, rather strangely the sentence seems to have been dropped from the second edition (Blaha & Rumbaugh 2005) but I still think it is just as true. I hope to make the process slightly easier for you by not looking at all the subtleties that Rumbaugh initially suggested. Instead we will concentrate on understanding the basic ideas and how to apply them.

Before considering what dynamic modelling is exactly, I will describe where it fits into the whole modelling process.
3.1 Where Does the Dynamic Aspect Fit in the Modelling Process?

One possible process of developing a dynamic model is shown in the diagram below. It should be noted that both OMT and UML make a point of stating that they are not ‘methods’. However, there are numerous modelling methods around that consultants often charge ridiculous amounts of money for giving you the privilege of using, the table below lists a few.

UML just provides the modeller with a set of diagramming tools which she/he is free to use or abuse. In contrast, often modelling techniques consist of very stringent sets of rules as well as diagramming tools. For example, one popular method called SSADM contains over 200 steps. While there are a huge number of methods now available it must be remembered that none of them have been tested rigorously by way of anything resembling a RCT (Randomised Controlled Trial); unfortunately some people argue that they do not require such rigorous evaluation processes.

As with all models, you can develop a dynamic model to describe a proposed system, a current system or help plan improvements. But we are running away with ourselves. Let’s start by considering a very useful and easily understood diagram – the Use Case diagram.

### A few examples of systems development methods that make use of UML

<table>
<thead>
<tr>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unified Software Development Process - USDP</td>
<td>Bennett et al 2004 p.11</td>
</tr>
<tr>
<td>Etc. etc</td>
<td></td>
</tr>
</tbody>
</table>
4. Use Case Diagram

Use Case diagrams provide an overview of the model you are developing. They show the main processes (‘use cases’) along with the things (‘actors’) that may interact with them.

There are numerous sites providing some excellent tutorials on UML, and as a demonstration of part of one I have included a section below from Practical UML: A Hands-On Introduction for Developers by Randy Miller (http://bdn.borland.com/article/0,1410,31863,00.html). I hope you like the medical nature of the example.

Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how.  [I would disagree. RB]

Use case diagrams are closely connected to scenarios. A scenario is an example of what happens when someone interacts with the system, in this example the system is a medical clinic.

"A patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot."

A use case is a summary of scenarios for a single task or goal. An actor is who or what initiates the events involved in that task. Actors are simply roles that people or objects play. The picture below is a Make Appointment use case for the medical clinic. The actor is a Patient. The connection between actor and use case is a communication association (or communication for short).

Actors are stick figures. Use cases are ovals. Communications are lines that link actors to use cases.

A use case diagram is a collection of actors, use cases, and their communications. We've put Make Appointment as part of a diagram with four actors and four use cases. Notice that a single use case can have multiple actors.

Use case diagrams are helpful in three areas.

- determining features (requirements). New use cases often generate new requirements as the system is analyzed and the design takes shape.
- communicating with clients. Their notational simplicity makes use case diagrams a good way for developers to communicate with clients.
- generating test cases. The collection of scenarios for a use case may suggest a suite of test cases for those scenarios.
Digging Deeper: Use Case Diagrams

Use case diagrams give an outsider's view of a system. Every use case diagram has actors, use cases, and communications. A simple use case diagram can be expanded with additional features to display more information.

This page covers the following UML™ use case features.

- system boundaries
- generalizations
- includes
- extensions

Medical clinic diagram, expanded

The following use case diagram expands the original medical clinic diagram with additional features.

A system boundary rectangle separates the clinic system from the external actors.

A use case generalization shows that one use case is simply a special kind of another. Pay Bill is a parent use case and Bill Insurance is the child. A child can be substituted for its parent whenever necessary. Generalization appears as a line with a triangular arrow head toward the parent use case.

Include relationships factor use cases into additional ones. Includes are especially helpful when the same use case can be factored out of two different use cases. Both Make Appointment and Request Medication include Check Patient Record as a subtask. In the diagram, include notation is a dotted line beginning at base use case ending with an arrows pointing to the include use case. The dotted line is labeled <<include>>.

An extend relationship indicates that one use case is a variation of another. Extend notation is a dotted line, labeled <<extend>>, and with an arrow toward the base case. The extension point, which determines when the extended case is appropriate, is written inside the base case.

End of extract
The above example may have left you with the impression that actors are people; however this is not necessarily the case. Let’s consider some examples:

<table>
<thead>
<tr>
<th>System you’re trying to model</th>
<th>Possible non person based Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>You are modelling the company project management department, and discover it has links with the main accountancy system for the firm</td>
<td><img src="main_accountancy_system.png" alt="Diagram" /></td>
</tr>
<tr>
<td>A departmental clinical system where you wish for it to have links to the internet</td>
<td><img src="internet.png" alt="Diagram" /></td>
</tr>
<tr>
<td>A purchasing system within a store where clients can pay by credit card as well as in cash</td>
<td><img src="credit_card_system.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

Also from the above use case examples you may have been thinking that the ‘actors’ could possibly be classes and the various ‘use cases’ in the Use Case Diagram (sorry about the terminology) could possibly be operations in particular classes. This is partly correct; it is better to think of them as candidate classes or candidate operations, only after further thought will you be sure. One thing you will find is that several actors may map onto a single class; this is because an actor describes a particular role. Take for example the above use case diagram; both the scheduler and clerk roles may well be amalgamated into the class secretary, with a title attribute indicating what job she/he actually does.

The term actor is rather misleading, and according to Fowler 2004 the term should be ‘role’. The problem was that the word was mistranslated from the Swedish (p100). One actor can be a specialised role of another actor, for example phlebotomist might be a specialised role of nurse:

The term scenario is rather nebulous. It is best to think of it as being a sequence of steps describing the interaction between actor(s) and one or more use cases in the Use Case diagram. In fact most case tools require you to provide a behaviour specification for each use case, which is basically a list of steps that occur within the particular use case. For example the use case Make Appointment could contain the following steps (behaviour specification):

1. The patient calls the clinic.
2. The patient gives the scheduler ID details for verification.
3. The scheduler checks validity of patient.
4. The scheduler checks system to see if a regular appointment time is planned.
5. The scheduler checks system for next available empty slot, depending on the type of patient.
6. The scheduler discusses options with the patient.
7. The scheduler books an appointment.
8. The scheduler logs the call in the patient’s records.
You can also organise a number of use cases into a group by using the UML package element.

UML 2.0 has introduced several new aspects to UML diagrams, one being the possibility of introducing \textbf{conditions} into the use case. The condition is shown as a comment on the diagram with a line from it to the relevant part of the extend line. For example suppose that employees of the company get their health care bills paid for them, otherwise the insurance company pays.

Several of the above examples are rather complex, including packages and extension conditions, and I feel the inclusion of all this complexity rather defeats the object of these diagrams. They are meant as simple overviews of what is happening. Including this type of complexity means there is a danger that you will do all the modelling at the use case stage. I think the success of a Use Case Diagram is by remembering:

\textbf{Keep it simple}

There are Sequence and State [machine] Diagrams for modelling the detailed dynamic complexities and we will start this process by revisiting the dynamic aspects of classes / objects.
5. What is the Dynamic Aspect?

'Well, here's another nice state you've gotten me into.' (Stan Laurel 1890 - 1965 mis-quoted)

Review of object syntax in UML (UML 2.0)

The top compartment shows the name of the object and its class. The object name should start in lower case and have no spaces. You should show the class name of which it is an instance using the syntax: object_name : class_name. All should be unlined.

The second compartment shows the attributes for the class as a list.

The current value of the attribute should be shown using the syntax: attribute_name : type = value

Most of the above is optional. The minimal amount of detail is the object name : class_name.

Example of various ways of showing a object in UML (2.0)

<table>
<thead>
<tr>
<th>smith:Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID=478678</td>
</tr>
<tr>
<td>firstName=John</td>
</tr>
<tr>
<td>age=45</td>
</tr>
<tr>
<td>developmentalStage=2</td>
</tr>
<tr>
<td>hairColour=gray</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>smith:Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Person</td>
</tr>
</tbody>
</table>

or

Note the underlining

Called an 'anonymous' object.

Notice the :

It is important to realise that when discussing the dynamic aspect, we tend to focus at the instance (object level) rather than the class level. This is because we tend to discuss real world examples rather than a set of possibilities. For example rather than considering all the possible interactions between PATIENT and DOCTOR classes, we would rather consider specific interactions between several instances, for example Patient:John Doe and Doctor:Smith. This is particularly true at the start of the modelling process; only later on a great deal of consolidation takes place to incorporate the different possibilities, highlighted by analysing the numerous scenarios etc.

Dynamic modelling initially focuses on analysis at the instance (object) level

Because of this, from now on this document will refer to an object instance of a class simply as an object.
The dynamic aspect of an object can be thought of as that aspect which is related to time. Basically only three things can happen to an object over time:

- It can receive communications (messages) from other objects or itself.
- It can send messages generated by its operations.
- It can change its state.

In other words, the dynamic aspect is concerned with messages and states over time (see the diagram opposite).

Consider this. The person object receives a message from another object (or from itself). Consequently this message may affect the object or not depending upon the message and the state of the receiving (target) object. For example, the message may be 'grow up' which might change the value of the data item 'developmental age' depending upon the 'age' of the object. Other messages may not depend upon the various values of data items within the target object. One such message might simply be 'kill' which would have the same affect on the target object regardless of its state.

Because messages often act as external stimuli on one or more objects they are usually called **events**. These events then often change the state of the object(s) receiving them.

It should be noted that an object can send a message to itself.

**Exercise 1. Defining Activities**

- **Time**: 30 minutes maximum

1. Define some of the activities that you carry out at work. (Remember work can be paid or unpaid such as volunteering!) What, if any, messages might be required to act as stimuli to these activities? What object might be the originators of such messages?

2. Define some of the activities of a community nurse and GP. What, if any, messages might be required to act as stimuli to these activities? What object might be the originators of such messages?
5.1 Messages/Events

The concept of an event is given several different names by different authors, and conversely the same name has slightly different technical meanings for the cognoscenti depending upon the context. However, for our discussion we will assume such subtleties do not exist and assume the following are equivalent:

Event = message = stimulus

Events can be said to have the following characteristics:

- They happen at a point in time (e.g. button pressed).
- A 'time stamp' is passed to the target object because the event happens at a point in time.
- Usually there is no significant duration for the event. (This is often a bone of contention among those modelling a system.)
- They often pass information (data item(s)).

Let’s consider an example:

```
smith: Patient
  iD=27474
  age=34

Event = Message:
  name, data items

atEndofCorridor:Theatre
  status=can_accept
  full=no

To_theatre (patient_ID, Theatre_ID)

Target object
```

In the above diagram the patient object sends a message (event) to a theatre (target) object providing information about a particular patient, theatre and implicitly time. Looking at the value of the attributes in the theatre object (the bottom section of the rectangle), it can be seen that in all probability the theatre object either accepts the patient or rejects the patient.

The above depiction is very similar to a particular UML diagram type called a Collaboration diagram. I will not be discussing this type of diagram any more in this document as it is very similar to the UML Sequence Diagram, which we will be looking at in depth next. If you are interested in Collaboration diagrams you can find a good example in Pender 2002 p 187-202, or the System Architect case tool help file.

**Naming messages**

This is in fact a problem. Most authors use the name to reflect the activity required by the receiving object. So in the above example the message name would be something like, accept_patient, on the other hand Rumbaugh (always in the first edition, only for asynchronous messages in second edition i.e. p.138 and p.153) uses the message name to reflect the action that the originator of the message is undertaking. Considering the above example the message name might be go_to_theatre. For the rest of this section I will use the second approach. You should always state which method you are using.

After you have completed the exercise below we will move on to look in depth at the Sequence Diagram.

**Exercise 2. Naming Messages**

**Time:** 15 minutes maximum

Considering the previous exercise, suggest some data items that might be passed in some of the messages defined.
6. Sequence Diagrams

**UML 2.0 infrastructure definition:**

A diagram that depicts an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding event occurrences on the lifelines. A sequence diagram can exist in a generic form (describes all possible scenarios) and in an instance form (describes one actual scenario)...

Sequence Diagrams provide a graphical account of the order of events that might occur between the objects being modelled. They can be thought of again as scenarios. Consider the example below of a simple Sequence Diagram describing the interaction that might take place between particular instances of a patient, GP and a nurse.

The basic idea of a Sequence Diagram has been around for some time, and as the UML reference manual (ver 1.3 p306) states, "Much of this notation is drawn directly from the Object Message Sequence Chart (POSA) notation of Buschmann, Meunier, Rohnert, Sommerlad, and Stal (1996), which is itself derived with modifications from the Message Sequence Chart notation." Jacobson 1995 calls them Interaction diagrams (p132).

In the diagram the three vertical lines (called lifelines) represent the three instances of objects in the system under study: Abdul Hussein is an instance of the class patient, Dr Amy Smith is an instance of the class GP and Nurse Derek Little is an instance of the class nurse. Instances are used instead of the more abstract classes because we are concerned with a particular scenario not general behaviour at this point in the modelling. There is no significance to the horizontal ordering of the objects. Most case tools allow you to manipulate the order to obtain the most aesthetic result.

The events (shown by message arrows) have names (this is not obligatory) and pass from one object (the sender) to another (the target). Up to UML 2.0 you could have a message with multiple targets; however it does not seem possible now (see UML 2.0 Superstructure p.428). Time moves down the event trace diagram, but the exact vertical distance between arrows is not important. However, you can show a time period between events as demonstrated in the above. Also various labels (such as timing constraints, descriptions of actions during an activation, and so on) can be shown either in the margin or near the transitions or activations that they label.

You can show the data that passes with the message by including it in parentheses after the message e.g. Followup_instructions(leaflet_id) or give_instructions(urine_sample, sputum_sample, faeces).

One interesting thing to note is that because we are dealing with instances you can show messages between two instances of the same class. For example if you wanted to show two doctors conferring you might have Amy Smith:Doctor and Steve Wright:Doctor as two instances.

The diagram is a very simple example and might pass as a rough draft. One would expect a much more complex diagram to develop over time using a wider range of UML symbols. Also you would not have the labels explaining the various parts of the diagram in the final version. We will now look at developing this diagram by way of considering the various elements of a Sequence Diagram.
6.1 Frames

These are new to UML 2.0. Each Sequence Diagram should be within a frame. Within the left hand corner of the frame there is a title along with the keyword SD indicating that it is a Sequence Diagram. Here is an updated version of our initial SD with a frame added. You may think this is rather trivial at the moment, but later on it will become clear how such a simple addition to the semantics of the diagram can allow you to model more complexity.

6.2 Lifelines

Each lifeline, a dashed vertical line with a rectangle at the top indicating the instance name and class, participates in one or more interactions in the SD. The length of the lifeline represents the existence of the instance. If the instance is created or destroyed during the period of time shown on the diagram, then its lifeline starts or stops at the appropriate point; otherwise, it goes from the top to the bottom of the diagram. An object symbol is drawn at the head of the lifeline.

If the instance is created during the diagram, then the arrow, which maps onto the stimulus that creates the instance, is drawn with its arrowhead on the object symbol. If the instance is destroyed during the diagram, then its destruction is marked by a large “X,” (the stop symbol) either at the arrow mapping to the stimulus that causes the destruction or (in the case of self-destruction) at the final return arrow from the destroyed instance. An instance that exists when the transaction starts is shown at the top of the diagram (above the first arrow), while an instance that exists when the transaction finishes has its lifeline continue beyond the final arrow.

Enough of the theory. Let’s consider an example. Imagine that you're in Star Trek working on the Enterprise, and Nurse Derek Little is a hologram which you can create and destroy at will. You would represent this by moving the object name down to where he is created and use the ‘X’ (stop) symbol to show his destruction.

The lifeline can also possess other characteristics such as a line showing when it has ‘focus of control’ (called execution of control in UML 2.0) or if it is an active object. These are objects that can carry on their own thread of control without the need for receiving messages constantly.

I tend to ignore these semantic enrichments to the SD. So now let’s move on to the next aspect of the SD.
6.3 The Message

Some aspects of the SD have changed significantly with UML 2.0, not all for the good I feel. The diagram now has much greater semantic richness but many of the new approaches are less intuitive. Gone is the useful method of numbering messages in a particular SD (although it remains in other UML interaction diagrams). Because of this loss of some features which modellers have found useful, various writers such as Fowler 2004 (p.60) suggest that people develop UML diagrams that are not always completely standard. I would agree with this as long as the non-standard semantics are clearly explained. Let’s now consider various aspects of the message.

6.3.1 Message Types

So far in this document I have shown messages with an arrowed line. UML provides a number of different line styles to indicate different types of message.

The following arrowhead variations may be used to show different kinds of communications:

<table>
<thead>
<tr>
<th>Arrowhead Variation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filled solid arrowhead</td>
<td>This type of message requires a response. The sending object is unable to continue until one is received. For example the Dinner_ready message sent from you, having spent time preparing it, would stop you doing anything until the resulting activity of the object who received the message completed the action evoked by that message (i.e. come and sit at the table). It is said to be <strong>Synchronous or Procedural</strong> communication and if you were a programmer you might call the message a <strong>procedural call</strong>.</td>
</tr>
<tr>
<td>Stick arrowhead</td>
<td>This type of message does not require a response and does not require the sending instance to wait. An example would be a notification message where the sending instance would just carry on with a sequence of actions once the message had been sent. In this instance the message Dinner_ready send from you would not make you wait around; you might start the activity washing up or even eating! In versions of UML prior to v1.4 it was shown as a half arrow thus: It is said to be <strong>Asynchronous</strong> communication.</td>
</tr>
<tr>
<td>Dashed arrow with stick arrowhead</td>
<td>This indicates a creation of an object instance to which the message is sent. It is new to UML 2.0.</td>
</tr>
<tr>
<td>Dashed arrow with solid arrowhead</td>
<td>This indicates a reply message. This is often omitted or the return value shown on the initial arrow. There are also other new types in UML 2.0, but I will ignore those. From the above you may had surmised that the actual name of the message is probably going to be either the operation to be invoked or possibly some other name in the case of Asynchronous communication. This is actually the case.</td>
</tr>
</tbody>
</table>

**Note:** In previous versions of UML it was possible to use the stick arrow head as an undefined type of message which was called a Flat message. Unfortunately this seems to have disappeared in the latest version.

Fowler 2004 p. 61 is again wary of all this complexity and suggests that people use anything but the asynchronous variety with extreme care.
Let’s now update our SD with mostly Synchronous messages, as I presume that each instance will be waiting for a response from the other during each of the activities. Please take note that I have omitted the Reply messages.

### 6.3.2 Slanted Arrows

Usually the message arrows are laid out horizontally in a Sequence Diagram; however if you want to indicate that a particular message may take some time to complete its task the arrow may be slanted downward so that the arrowhead is below the arrow tail.

The SD opposite (Tsang, Lau & Leung 2005 p.162 simplified) shows the use of such lines along with the non UML 2.0 use of message sequence numbers, and the lack of a frame!
6.4 Showing More Detail – Interaction Occurrence

UML 2.0 has introduced the concept of 'InteractionOccurrence' – no space (UML 2.0 superstructure p.423). An interaction occurrence indicates a number of interactions (called an Interaction fragment in UML 2.0 speak) which occur in another diagram. This is indicated by using a frame which is drawn to cover the lifelines involved in the interaction fragment and you can define parameters as well as a return value.

This is excellent because it means that you can move detail to several smaller Sequence Diagrams, each representing an interaction fragment, and reuse these smaller components. Considering our example:

Frames also have other uses including specifying how interactions are controlled, a specification method new to UML 2.0.

6.5 Control

Let's think how the flow of control in a SD might be affected:

- We might want a particular interaction fragment to be repeated until a certain condition is achieved = \textit{loop(ing)}
- We might want a particular interaction fragment to take place only if a specific condition is met = \textit{opt(ional)}
- We might want alternative interaction fragments to take place depending upon a particular condition = \textit{alt(ernative)}
- We might wish several interaction fragments to go on at once = \textit{par(allel)}

Each of the highlighted words in the above sentences indicates the keyword that UML uses to specify the particular condition you are modelling. Let’s look at each of these in turn.

6.5.1 Iteration

In our SD we have the message *for all samples*:supply_sample. This is an example of an iteration; an interaction fragment repeats until a certain condition is achieved.

Iteration is just another name for repetition / looping; it indicates that a particular message repeats a certain number of times. The number may be specified or be related to some condition or even initially be unspecified.

How we show this in a SD has changed with UML 2.0. Because the two methods are so different and most case tools still use the older method, I will provide details of both.
Pre UML 2.0

An iteration is represented by an asterisk (*) followed by an expression in square brackets followed by the message name. For example:

* ([While items in stock]: buy
* [For each item]: catalogue
* [While not unconscious]: drink
* [For all samples]: supply sample (as in the above Sequence Diagrams)
* [While hospital not full]: admit
* [i := 1..n]: send message (taken from the UML specification v1.4 p3.133)

If you are not sure of the iteration condition you can miss it out and just have the asterisk followed by the message name e.g. * admit, * update etc. The UML v1.4 specification is unclear about the use of the asterisk, mentioning it in the narrative v1.4 p3-133 and expression example on p2-134 but not showing it in the example Sequence Diagram on p3-104.

UML 2.0

This uses the frame element once again, showing the keyword loop and the condition at the top. An example may help.

Obviously this example is trivial with just one message, but you can easily see how you can specify several. Unfortunately the case tool (VP-UML) has a bug and hides the life lines within the loop frame.

You should also notice that the message has just been split into two halves: the expression and the message name.

6.5.2 Single Branching

Suppose in our example we wish to show that given a particular condition (called a guard condition), a certain interaction fragment is executed. Again there is the old and new way of doing it. To be more specific, we only want the interaction fragment to be executed if the guard condition evaluates to true.

Pre UML 2.0

The guard condition is represented as an expression within square brackets followed by the message name that will take place when the expression is true. As in the case with iteration, the expression and message would appear on the appropriate message.

Here are some examples:

[if patient weepy]: give reassurance
[product available=yes]: buy it
[Finished your vegetables]: leave the table
[balance>0]: borrow money

In UML pre 2.0, message lines could divide to indicate conditional branching.
Back again to the use of a frame, this time with the `opt` keyword, all the other aspects mentioned above for the loop situation also apply here. Imagine that you feel you are giving some patients too much time and really only want to reassure those that you feel really need it, such as those who appear a bit weepy:

The frame now has three compartments, called operands, and the frame is called a combined fragment. Finally each of the conditions is called a condition-clause. However, as usual it’s more important to be able to use these constructs correctly rather than just know the name of them.
6.5.4 Concurrent Messages

Concurrent messages are the last type of message I want to describe. In this situation you want to be able to model the situation where more than one thing is happening at once. UML 2.0 allows you to do this by once again using a frame but this time with the par keyword. An example should suffice. Assume now that holographic nurse Derek can carry out several activities at once.

It should be noted that the SD tells us nothing of the timing of each of these operands, only that they will run concurrently; we do not know which, if any, may finish first etc.

6.6 Nesting

So far we have not seen one interaction fragment within another, but this is quite possible. See the SD opposite.

I’m sure you will be pleased to know that you have now learned the basics of Sequence Diagrams, but the important thing is to get as much practice as possible. To help you achieve this I have provided a number of exercises on the next few pages. Just a quick word of advice first.

6.7 Incremental Development

In the process of modelling, the modeller gradually develops a number of scenarios considering first how the system would behave in normal conditions and then when something untoward might happen (e.g. the patient passes out or the nurse is not available). Finally the modeller consolidates them by comparing and contrasting each scenario. UML provides a special diagram to allow you to amalgamate them without getting too much clutter: the Interaction Overview Diagram. However I feel that you can achieve a similar result by making use of the ref frame as we did earlier. Now onto some practice for you.
6.8 Exercises - Sequence Diagrams

Exercise 3. Improving Sequence Diagrams

Below are given some draft Sequence Diagrams. See how many errors you can spot and make as many improvements as you can to make them into UML 2.0 compliant Sequence Diagrams.

A. A Restaurant

```
Restaurant

waiter  customer  kitchen

<table>
<thead>
<tr>
<th>bring menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>discuss options/specials</td>
</tr>
<tr>
<td>read menu</td>
</tr>
<tr>
<td>Place order verbally</td>
</tr>
<tr>
<td>write down order and numbers of items</td>
</tr>
<tr>
<td>place order list and number of items</td>
</tr>
<tr>
<td>prepare food</td>
</tr>
<tr>
<td>food ready to be served</td>
</tr>
<tr>
<td>food is served</td>
</tr>
</tbody>
</table>
```
B. Organising a Local Medical Conference

Another rough draft for you to work on.

<table>
<thead>
<tr>
<th>Delegates</th>
<th>Speakers</th>
<th>organisers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form organising committee</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decide on topics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select possible speakers</td>
<td></td>
<td>Select venue and reserve</td>
</tr>
<tr>
<td>Invite speakers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send out first announcement</td>
<td>collect replies of interest from public</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send out second announcement and registration</td>
<td></td>
</tr>
<tr>
<td>Collect registrations</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Make travel accommodation arrangements</td>
<td></td>
</tr>
<tr>
<td>Hire venue and AV equipment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
C. Carrying Out Cerebral Angiography (only attempt this one if you are a medical doctor or nurse who might want to annotate it significantly!)

<table>
<thead>
<tr>
<th>Radiologist</th>
<th>neurosurgeon</th>
<th>patient</th>
<th>nurse</th>
</tr>
</thead>
<tbody>
<tr>
<td>receive request for angiography</td>
<td>check indication with neurosurgeon</td>
<td>review patient’s CT/MR scan</td>
<td>introduce self to patient</td>
</tr>
<tr>
<td>explain procedure</td>
<td>reassure</td>
<td>obtained informed consent</td>
<td>check BP &lt;160 systolic &lt;120 diastolic</td>
</tr>
<tr>
<td>check INR is 1.2 or&lt;</td>
<td>IV drip</td>
<td>order diazepam 10mg orally 2hrs prior</td>
<td>book anaesthetist</td>
</tr>
<tr>
<td>book angiography suite</td>
<td>check on patient in ward 2hrs prior to procedure</td>
<td>perform 4 vessel angiogram</td>
<td>read angiogram</td>
</tr>
<tr>
<td>report angiogram</td>
<td>discuss report</td>
<td>check puncture site for 6 hours in ward</td>
<td>check BP and pulse for 6 hours post procedure</td>
</tr>
</tbody>
</table>

Many thanks to past students for providing the basis for the above examples.
D. A Published Example (see how many errors you can spot and think of ways to possibly improve it)

From: http://www.visual-paradigm.com/product/vpuml/htmltutorial.jsp
Exercise 4. Developing Your Own Sequence Diagrams

**Time:** 60 minutes maximum

1. Devise a Sequence Diagram for two situations (scenarios) involving yourself at work.

2. Devise a Sequence Diagram for **one** of the following situations:
   - Planning a 6 month holiday
   - Building or renovating something
   - Items of service claims forms for GPs
   - Clinical audit within a hospital department
   - Multidisciplinary assessment in the community
   - Commissioning negotiation
   - Accident and Emergency assessment
   - Care programme approach review meeting in psychiatry


Space left for you to make notes:
7. Business Process Re-engineering (BPR)

This very impressive term is really nothing more than analysing an organisation using – amongst other techniques – Sequence Diagrams. But as would be expected, they are given numerous other names in BPR. BPR uses the Sequence Diagram concept to consider how the events in an organisation might be re-organised for more efficient and/or effective behaviour. I feel an example coming on:

A particular health authority are running a project called 'heartrisk'. This is a GP community based project which provides data concerning individual patient risk of coronary heart disease based upon measuring various risk factors (smoking, drinking, weight etc.) to patients in the form of a hand-held sheet. At the present time a representative visits practices they feel might be interested in participating in the project. If they are interested, a representative from the external software company is informed who in turn informs a colleague who visits the GPs and sets up the GP system to collect the relevant data. A representative of the software company or the health authority then visits the site to train them. A further visit by a member of the software firm collects a copy of the GP system backup tapes to obtain the necessary data after a suitable time period. Following initial data analysis by the software firm in consultation with the practice, a ranked list of each patient's CHD risk score is printed out and sent to the practice. The practice then decides who would most benefit from the leaflets and requests them from the software firm. These are then posted out to the practice which contacts the patients to offer them advice if they haven't died already!

The options available to improve this scenario are numerous. Let's consider the one simple strategy of allowing the GPs' own system to do what the external software firm now do to a limited extent. The new event trace is given below. It immediately looks much less complex. Event traces are extremely useful for analysing situations such as the one above.

(Business) Process Re-engineering is a very important aspect of modelling which is frequently ignored by people not involved in management. This is a shame as such people can bring a degree of analytical rigor which is often lacking. For more information about BPR see the links section in this document.

Interestingly the NHS some while ago produced a document about BPR: “Management of Change: An Overview of Management Change Methodologies” (NHS E 1997 ref no.D4054). Unfortunately the document is very thin on information and basically just provides a list of methodologies.

We will now return our attention to UML and specifically to the analysis of the dynamic aspect of an individual class.
7.1 Relationship Between Sequence Diagrams, Use Cases and Class Diagrams

In passing we have mentioned that there is a correspondence between the instances shown in the Sequence Diagram and the classes in the class diagram. We have also mentioned the possibility that certain messages (usually synchronous messages) may be mapped to particular operations in a class. Such possible operations are called **candidate operations**.

The exact method depends upon the style of message naming you have adopted. We will first consider the originator naming style as used above.

7.1.1 Originator Message Naming Style

So if we are using the originator style message naming:

**Messages originating from instance = candidate operations for that class**

Now let's consider the more common message naming style.
7.1.2 Target Style Message Naming

Some possible operations for classes, derived from the Sequence Diagram

<table>
<thead>
<tr>
<th>Patient</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
</tr>
<tr>
<td>ProvideHistory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>..</td>
</tr>
<tr>
<td>start_Examine</td>
</tr>
<tr>
<td>ProvideCuddle&amp;reassure</td>
</tr>
<tr>
<td>ProvideReassure</td>
</tr>
<tr>
<td>GiveHandout</td>
</tr>
<tr>
<td>ToNurse</td>
</tr>
<tr>
<td>Provide_results</td>
</tr>
</tbody>
</table>

So if we are using the originator style message naming:

Messages pointing towards instance = candidate operations for that class

7.1.3 Use Cases

Actors are a bit more problematic. Frequently they represent several instances of one particular class, for example various roles a nurse might undertake. This indicates that several actors may become one class. The individual use cases themselves occasionally become operations, but they are usually too high level and require further refinement to map to low level operations. Tsang, Lau and Leung 2005 p.191 provide a little more detail, indicating that the single use case specification (i.e. the list of steps within a single use case) can be mapped to a series of messages between instances in a Sequence Diagram which then themselves become operations in various classes. Obviously there is nothing stopping you defining operations in a class and then making use of them in a SD – which way round you do it is up to you. I tend to ignore the Use Case diagram after producing a very simple overview. Basically we have:

Actors  →  Classes

Steps within a use case or sometimes actual individual use case  →  Message (event) in SD  →  operation in a class or event that stimulates it
The diagram below indicates the relationship I have described above. Now you possibly can see why it is advantageous to use a case tool which keeps all the diagrams synchronised rather than a simple drawing tool.

**Exercise 5. Relationship Between Sequence and Class Diagrams**

**Time**: 20 minutes maximum

1. Consider one of the Sequence Diagrams you have drawn and develop a class diagram from it. Make sure you include operations for the various classes derived from the messages in the SD.

Now let's move onto the next type of dynamic diagram: the State Machine.

So far we have focused on the interaction between different instances, usually between different classes. Now we will concentrate on the messages within a single Class by investigating the UML diagram used to model this aspect.

The diagram UML used to model the dynamic nature of individual class is the State [Machine] Diagram, often referred to as state diagrams. State diagrams (called the same thing in OMT and UML) have been used in a number of forms for well over two decades. OMT and now UML have added an array of additional symbols to the diagrams to provide a greater degree of expression. However, we will only consider the basics in this section.

State diagrams show visually the relationships between transitions and states for a single class. In this section will look at state diagrams in depth and then consider how a state diagram is developed. An 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. It also relates to the Consultation SD described earlier. Comparing the above state diagram with that of the SD using the target message naming convention (on page 31) we note that there are similarities; the messages (i.e. events) which are the arrows on the SD that point towards the GP instance appear to roughly equate to the states in the above diagram. Also events (messages) in the SD which neither originate nor terminate at the GP object such as introduce_self_to_nurse (in the nurse consultation interaction fragment) do not appear in the above diagram. Much more about this later.

The rounded boxes represent states, and the lines represent transitions. Transitions can be considered here as equivalent to events (i.e. messages) which we have already considered earlier in this document; however we have not yet considered states so let's now rectify the situation.

### 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 (object) level</td>
<td>At the class level</td>
</tr>
</tbody>
</table>

Robin Beaumont robin@organplayers.co.uk  17/10/2007  Page 33 of 47
8.1 States

A state is usually a result of a particular configuration of the object 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 sets of values that the data items for a particular instance have at a particular time.

A particular person (‘instance of class person’) is in a particular state at any one time. A state is defined by a particular set of values the data item possesses (e.g. age and developmental stage) or by a single data item (e.g. in the case of the hospital operating theatre which has a status data item with the value taking at least ‘usable’ or ‘being_decotated’).

Consider the examples below:

<table>
<thead>
<tr>
<th>Examples of some Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some Configurations for a person:</td>
</tr>
<tr>
<td><strong>Person</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>age</td>
</tr>
<tr>
<td>Developmental stage</td>
</tr>
<tr>
<td>Hair colour</td>
</tr>
<tr>
<td>Hair state</td>
</tr>
<tr>
<td>Doing shopping</td>
</tr>
<tr>
<td>eating</td>
</tr>
<tr>
<td>State</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Some Configurations for a HOSPITAL theatre:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Theatre</strong></td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Full</td>
</tr>
<tr>
<td>Accept_patient</td>
</tr>
<tr>
<td>Reject_HIV_patient</td>
</tr>
<tr>
<td>State</td>
</tr>
</tbody>
</table>

States can be considered to fall into two main types: idle or active (i.e. performing some type of activity). Therefore states can be thought of as actions or activities some of the time.
Both OMT and UML take this idea much further, dividing up the concepts in a rather complicated way which I will not consider here. Now we have considered states and transitions separately, let's consider their relationship, but first an exercise for you!

Exercise 6. 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 object which can take a number of values. For example, considering a banana, its state might be {growing, ripe, going off}, resulting from various configurations. Can an object 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 banana?

8.2 Relationship between Transitions and States

The life of an object is basically the movement through a number of states, each being nudged along by a transition.

So far we have considered events and states separately but from the diagram below there is an obvious relationship. Look at the diagram.

Notice that in the diagram opposite I show an event on each transition line.

Events in state diagrams usually represent things that are very clear-cut such as:

- Goods dispatched
- Consultation ended
- Card arrested
- Card rejected
- Start operation
- Fan off
- Select
- Die
- etc

Exercise 7. Drawing State Diagrams

**Time:** 30 minutes maximum

1. From the list of roles/states you developed in the last exercise, think of possible events 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 an event that did not change a state? [Hint: think about recursion.]
8.3 Condition Statements

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 33) where an event name occurs on most of the transition lines indicating that the transition is fired off by the event. For example the event Patient_arrives triggers the transition to examining etc.

8.4 Initial and Terminal States

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

You should realise that the first state is the one that the arrow point 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!
8.5 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 SD we created earlier.

8.6 Other State Compartment Options

A state may optionally be subdivided into multiple compartments separated from each other by a horizontal line as in the above. The main compartment called the internal transitions compartment can contain any number of several types of activities, each indicated by a keyword:

- **entry**
  This label identifies an action, specified by the corresponding action expression, which is performed upon entry to the state (entry action).

- **exit**
  This label identifies an action, specified by the corresponding action expression, which is performed upon exit from the state (exit action).

- **do**
  This label identifies an ongoing activity ("do activity") which is performed as long as the element is in the state or until the computation specified by the action expression is completed (the latter may result in a completion event being generated).
• include
This label is used to identify that another state diagram is included here.

8.7 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.
9. Obtaining an Initial State Diagram from a Sequence Diagram

The task of developing a state diagram from a SD 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:

This assumes you are using the target style message naming:

1. Consolidate the Sequence Diagrams.
2. Take an individual object in the SD and:
   - Consider those (message) events for which it is a target (arrows towards it), and consider these provisionally as **states** for the object. (Target style naming used in diagram)
   - Consider those (messages) events for which it is the originator (arrow away from it), and consider these provisionally as **events** for the object.

Looking back once again at the SD 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 Doctor class:

- examining
- reassuring
- Sending to nurse
- Providing results
Exercise 8. Converting Sequence to State Chart Diagrams

**Time:** 15 minutes maximum

Using one of the SDs you developed earlier, take one of the objects and create a state diagram.

Remember the appropriate rules given above concerning which events on the event trace should be considered to be states or events within the state diagram. Notice that you may want to enhance the state diagram once you start to look in more detail at the individual object. This is quite normal and to be expected.

10. Importance of Dynamic Modelling

In the above sections you have learned a lot about dynamic modelling and I’m sure you must have a headache now!

What is so amazing is that dynamic modelling has often been ignored in health situations in the past. However it is becoming more and more important as the realisation that understanding how people work and how this will change with the introduction of computers is still poorly understood in the health service.

One area where dynamic modelling is being applied with more and more success is that of specifying 'business processes' within an organisation. The use of this technique was clearly demonstrated in the example concerning the 'hearthrisk' project. As mentioned at the beginning of this document, in 1997 the NHS Executive produced a book on Business Process Reengineering in the NHS. Unfortunately the book contains little in the way of practical advice. In fact, one feels it is trying to encourage the use of external consultants rather than encouraging people to learn the skills themselves.

Dynamic models are also used to describe computer interfaces. Each screen or window is represented as a state while the event between each state is the various mouse clicks or key presses required. This aspect is covered in a separate document which you will find at [http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm](http://www.robin-beaumont.co.uk/virtualclassroom/contents.htm).

Exercise 9. Developing Dynamic Models

**Time:** 120 minutes maximum

1. Consider the DopeHead scenario and draw a Sequence Diagram for a typical blood sample. Details of how to obtain the modelling scenarios document is given in the “Before You Start” section at the beginning of this document.

2. Select a particular aspect of your work and draw a Sequence Diagram. Some possibilities might include:
   - Patient admission
   - Patient discharge
   - Transferral to ITU
   - Preparation for an organ transplant
   - An outpatient clinic
   - If you do not work in the health field think up some of your own

You should then present your findings to your colleagues with suggestions for any enhancements that you think are appropriate. Take note of any comments they make about these proposed enhancements. Did you forget some important characteristics of the system when suggesting them?
3. Ask a friend about a particular scenario. Preferably, choose a profession unrelated to your own.

4. Looking at the modelling scenarios document choose one of the scenarios and attempt to model its dynamic aspect. Details of how to obtain the modelling scenarios document is given in the “Before You Start” section at the beginning of this document.

5. Using one of the SDs you developed earlier, take one of the objects and create a state diagram. Remember the appropriate rules given in the previous section concerning which events on the event trace should be included.

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. Which of the following best describes the dynamic aspect of system modelling:
   
   a. Investigation of data aspects  
   b. Investigation of temporal (time) aspects  
   c. Investigation of functional aspects  
   d. Investigation of security aspects  
   e. Investigation of general modelling aspects  

2. The dynamic aspect of objects is reflected in their:
   
   a. Data values  
   b. Actions  
   c. Attributes  
   d. Class names  
   e. Relationships  

3. Messages can be considered to be equivalent to:
   
   a. Warnings  
   b. Activities  
   c. Responses  
   d. Triggers  
   e. Communications  

4. The process of developing dynamic models involves, amongst other things:
   
   a. Development of Sequence Diagrams based upon identified classes  
   b. Development of Sequence Diagrams from event traces  
   c. Development of classes from event traces  
   d. Development of work improvement plans from BPR  
   e. Development of table definitions from identified classes
5. At a practical level which of the following terms best describes the 'event' concept:

a. Trigger  
b. Message  
c. Warning  
d. Activity  
e. Function

6. A Sequence Diagram is said to represent:

a. A lifeline  
b. A scenario  
c. A message board  
d. A state sequence  
e. Object relations

7. What is the name of the above diagram:

a. Event matrix  
b. Class inter-relationship diagram  
c. Sequence diagram  
d. Message board  
e. State diagram

8. What type of thing does "Label 1" point to in the above diagram? Note that this may be different from the standard approach taken to this type of diagram.

a. Class name  
b. Data sources  
c. Object instance  
d. Event source  
e. Actor
9. What is the correct title for 'Label 2' above:
   a. Event
   b. Trigger
   c. Warning
   d. Information arrow
   e. Data flow

10. What is the correct title for 'Label 3' above:
    a. Time
    b. Time in hours
    c. Time in days
    d. Scenario boundary
    e. Does not possess a sensible label

11. What is the correct title for 'Label 4' above:
    a. Lifeline
    b. Timeline
    c. Object existence boundary
    d. Dataline
    e. Does not possess a sensible label

12. What is the correct title for 'Label 5' above:
    a. Lifeline
    b. Time period
    c. Existence constraint
    d. Association
    e. Does not possess a sensible label

13. Which statement is true:
    a. OMT and UML are basically sets of diagramming tools.
    b. OMT and UML are sets of diagramming tools along with rules for software development.
    c. OMT and UML consist of a set of diagramming tools, software development rules and suggested management structures.
    d. OMT and UML are different in many ways.
    e. OMT and UML provide a small number of diagramming tools but a detailed method for software development.

14. When developing scenarios:
    a. Scenarios are developed gradually, encompassing more complex/exceptional circumstances. Eventually an attempt is made to consolidate them.
    b. Scenarios are developed for mutually exclusive situations which are eventually joined together.
    c. Scenarios of simple situations are developed and then consolidated.
    d. Scenarios are developed gradually, encompassing more complex/exceptional circumstances. Finally a single, all embracing one is produced.
    e. Scenarios are developed by the system developers to propose enhancements to manual systems.

15. BPR stands for:
    a. Business Purpose Realignment
    b. Business Purpose Redevelopment
    c. Business Process Reengineering
    d. Business Process Redevelopment
    e. Business Process Realignment
16. BPR makes extensive use of which type of diagram:

   a. Sequence  
   b. Event  
   c. Dataflow  
   d. Class  
   e. Higraph

17. What is commonly the main purpose of BPR?

   a. To consider how proposed events in an organisation might be organised for maximum efficiency and effectiveness  
   b. To radically reorganise an organisation for maximum efficiency and effectiveness  
   c. To consider how the events in an organisation might be reorganised for more efficient/effective behaviour  
   d. To consider how present or proposed events in an organisation might be reorganised for more efficient/effective behaviour  
   e. To monitor the current efficiency and effectiveness against a theoretical optimal level

18. States can be considered to fall into the following two main categories:

   a. Idle/active  
   b. Isolated/interactive  
   c. On/off  
   d. True/false  
   e. Alive/dead

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

   a. State diagrams show visually the relationships between events and states for several objects.  
   b. State diagrams show visually the relationships between events and states for a single object.  
   c. State diagrams show visually the events for a single object.  
   d. State diagrams show visually the states for a single object.  
   e. State diagrams show visually the relationships between events for several objects.

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

   a. Triggers  
   b. Alarms  
   c. States  
   d. Events  
   e. Initial conditions

21. 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
22. If you see the expression "completes examination [if weepy]" along a line in a state diagram, it would represent the following:

   a. Textual description (unnecessary)
   b. Message
   c. Condition statement
   d. Event
   e. Activity

23. What is the correct term for nesting?

   a. Inclusion
   b. Decomposition
   c. Induction
   d. Fragmentation
   e. Deduction

24. 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**

Dynamic modelling is gaining in importance as it becomes clear that often the problem with computer, as well as human, information systems is how they work and not just the data they produce.

The techniques of dynamic modelling are still relatively in their infancy compared to the traditional entity relationship (ER) modelling and its more modern equivalent, object modelling. This is probably due to a number of factors. Dynamic modelling has been traditionally less used by modellers so has not been so well developed, resulting in a catch 22 situation with fewer people using dynamic modelling because the tools are less useful because few people use them. There is also perhaps the fact that dynamic modelling is conceptually more difficult for the modeller compared to the sometimes almost somnambulant process of defining objects or entities.

In this document we have barely scratched the surface of dynamic modelling but probably have still gone further than most traditional courses, a worrying prospect given the dire consequences of not considering the dynamic aspect of a system.

The diagram opposite once again shows the main stages in developing a dynamic model.

Now return to the learning outcomes at the beginning of this document and see how many you can tick off. Remember that at the start of the document you were warned that this subject is difficult!
13. References

Also see the document concerned with class/object modelling.


Jacobson I 1995 The Object Advantage. Addison-Wesley.

NHS Executive 1997 Business Process Reengineering in the NHS. NHS executive UK

Pender, A Thomas 2002 UML Weekend crash course: 15 hours. Wiley

14. Links

For details of dynamic modelling and UML see the object modelling subsection. The following links relate to BPR.

Special Issue Impacts of Information Technology Investment on Organizational Performance Fundamental to the concept of BPR is the issue of automation: should you, and how much should you automate? This article provides chilling reading about the actual advantage of automation. (Click here for a local copy.)

BPR Business Process Reengineering Research, Tools, Practice This site from Warwick University provides not only a large number of useful links (warning: many of them are not working unfortunately) but also teaching material.

Business Process Redesign - An Overview (Article) This is an excellent introductory article which also explodes the myths concerning BPR. Well worth a read. (Click here for a local copy.)

Humanistic approach to BPR (+ bibliography from Nasa) A well referenced article about the managerial problems with BPR and how to cope with them.

Pender also writes a UML bible which contains the following resource page:


For a very complete list of case tools for UML see:

http://objectsbydesign.com/tools/umltools_byCompany.html

Add your own:
15. Appendix A – Technical notes

The UML 1.3 complete message specification

After all the options that have been presented over the last few pages you may be confused at to what the text about a message arrow actually means.

As you can see you can make a message rather complex. Note that the message name can be the same as one of the operations of the sending instance. I have taken the above from Pender 2002 p181 (adapted).