UML State Diagrams with Magicdraw

Robin Beaumont robin@organplayers.co.uk

Thursday, 11 October 2007

Contents

1. Introduction

1.1. Where to obtain the MagicDraw software

This depends on who you are:

- MSc students at Bath or The Royal College of Surgeons (Edin.) will be provided with the software along with the appropriate academic licence.
- All others can obtain the software by visiting and registering at: http://www.magicdraw.com/ for the free community edition. An alternative with a similar interface is Visual Paradigm (VP-UML).

1.2. Before you begin this tutorial?

Before you work through this tutorial you should have completed several other UML practical and pen based tutorials as this one is the last in the series. Specifically:

Practical tutorials	Knowledge tutorials
Introduction to ERD modelling using UML Class diagrams with Magicdraw	Starting to model using erds
Introduction to Class modelling using UML	Information modelling using UML 2.0 Classes
UML Use Cases using MagicDraw	An Introduction to Dynamic Modelling and Process Re-engineering Using UML
UML Sequence and Communication diagrams with MagicDraw	

All the above tutorials are available at http://www.robin-beaumont.co.uk/virtualclassroom/case_tutorials.htm

1.3. What are the aims of this tutorial?

This tutorial is the fifth and last in a series of practical tutorials to introduce you to using a specific CASE tool, MagicDraw Personal Edition (MD/PA). This tutorial assumes that you have worked through the first four tutorials so in several places it rather briefly describes what to do. By the end of this tutorial you will feel confident about using MD/PA to draw UML compliant State diagrams. You will notice that this is the third practical tutorial to teach you how to model behavioural (dynamic) aspects of classes/ objects emphasising the importance of this aspect.

2. Revision

It is important that you know what a State diagram is and how it relates to other UML diagrams before you work through this practical tutorial, some basic facts:

Pender 2002 p. 204 provides a nice succinct description of what a state diagram is which I have adapted slightly, The state diagram models the ***events*** that trigger a ***transition*** (change) from one ***state*** to another state **within a particular class**. Each event may have a corresponding ***activity*** that makes the changes in the class instance (that is alters the attribute values). While the class instance is in a state, it may also perform work associated with that state. Such work is also called an ***activity*** (also called a **do** activity). The state diagram can also be used to model concurrent activities within a state by creating possibly parallel ***substates*** within a ***superstate*.** Using the substate and superstate notation, you can explicitly identify split and merge control for concurrency.

Important points are:

- The state diagram models at the class **not** the instance level.
- The State diagram shows the events that cause a transition from one state of a class to another state **within a single class.** This is in contrast to a sequence diagram that shows interaction between class instances (objects).
- The scope of a State diagram is the entire life (in terms of states and events) of a single class, that is you can see all the states a single class, and hence any instances of it can ever be in. In contrast in a Sequence diagram it is usually only possible to show a subset of states within a particular scenario. Therefore:
- Several Sequence diagrams may be needed to derive a complete state diagram for a class (more about this latter).
- The System architect help file also makes an important point: Modeling a state diagram is useful for understanding the dynamic behaviour of *important* classes in the system. By important, we mean a class that performs important tasks and may change state often based on system or business events. What is also implied is that you do not need to build a state diagram for every class in the system, just ones you'd like to understand better.

Exercise 1

From the information provided in the above revision section and information in the knowledge tutorial "An Introduction to Dynamic Modelling and Process Re-engineering Using UML" Answer the following:

1. Draw below the symbol that is used in a state diagram to represent a state?

2. Draw below the symbol that is used in a state diagram to represent a transition?

3. How many sequences diagrams would usually be required to specify a complete State diagram?

4. A state diagram models at the instance level, true or false?

5. A state diagram always relates to several classes in a class diagram, true or false?

6. There are as many state diagrams a classes in a model, true or false?

3. Developing State diagrams two approaches

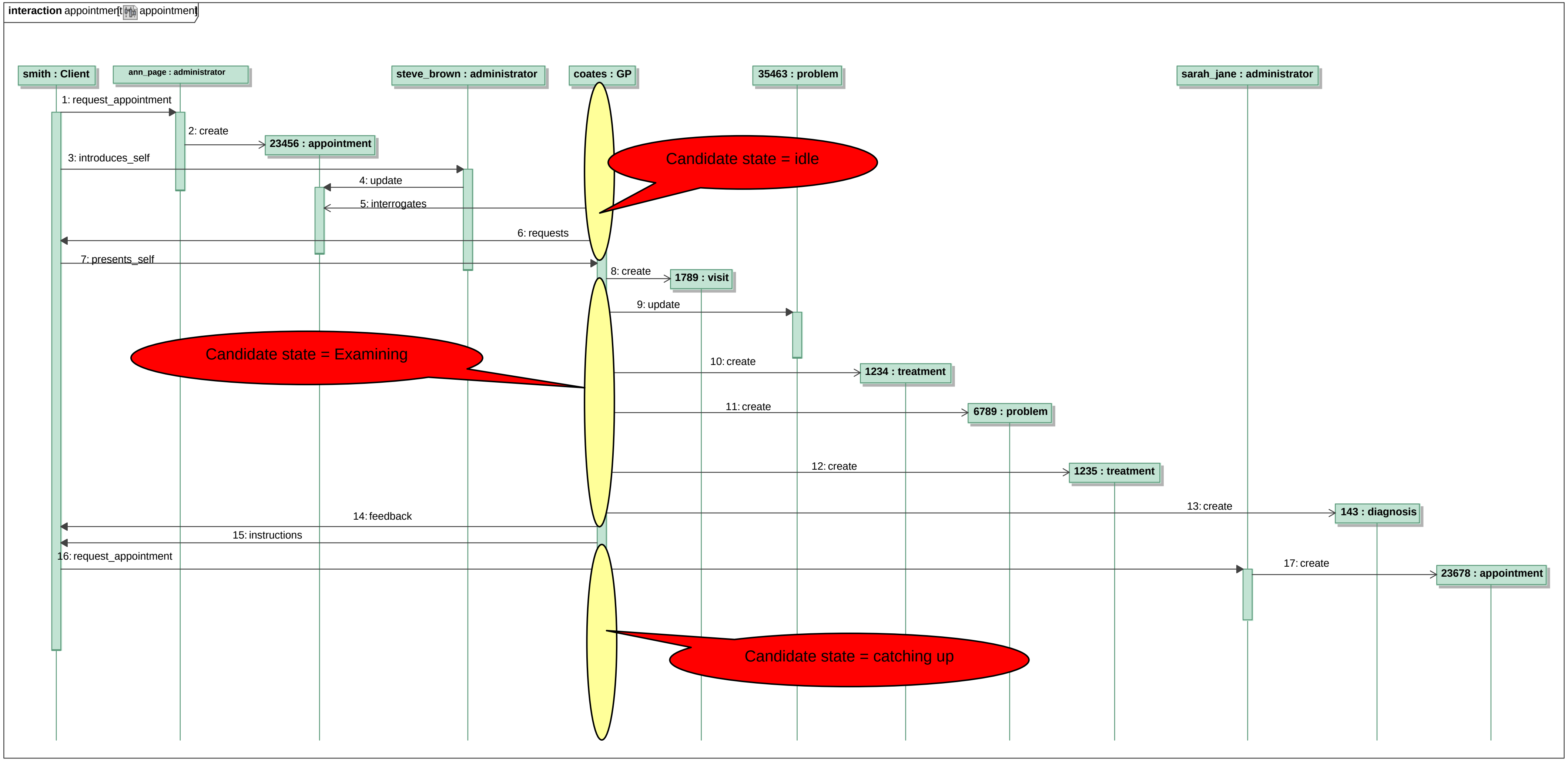
There are two principle ways to develop State diagrams:

- Develop them from narrative descriptions as we have done or the other UML diagrams.
- Derive the information partially from sequence diagrams already produced.

Because the previous practical UML tutorial created a sequence diagram I will use the second method, but mention how I have made use of the narrative description in the process.

4. Choosing a class to be the state diagram subject

Possibly the most interesting four classes concerning states, are the Client, Administrator, GP and Problem. I have chosen the GP. As described in the dynamic modelling knowledge tutorial (page 38) messages that **target** the instance, i.e. Dr Coates, can be considered as candidate state changes (transitions). Likewise, you can study the **outgoing** messages to find activities that the class carries out while in a particular State. You should realise that often return messages are not shown in sequence diagrams for example message 15, instructions I have assumed would send a return from the client to change the doctors state again.



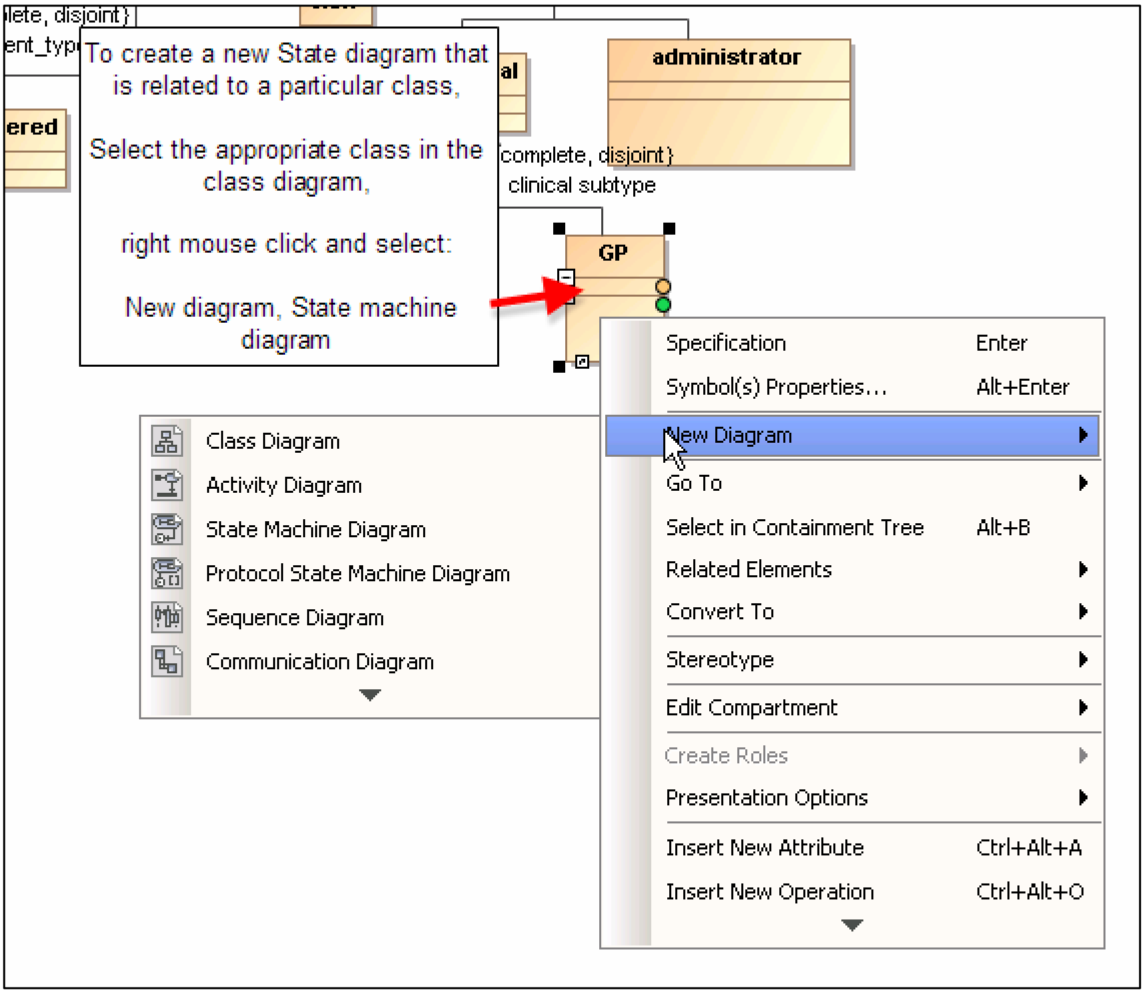
It is important to remember that another sequence diagram may model other, or the same, instances of the GP class which would suggest additional states, for example in the Dynamic modelling knowledge tutorial we have states such as , Reassuring, Sending to nurse, Providing results, as additional states for the GP class.

Often the sequence diagram does not provide much detail in terms of individual activities for instance, in the State diagram we can be much more detailed, discovering and capturing activities, particularly internal 'do' activities, not included in the Sequence diagram. For example the Candidate state consulting, clearly has several "do" activities such as Update problem (message 9), Create treatment (message 10), Create problem (message 11) etc. We can either model these as internal 'do' activities or as separate state transitions in the GP state diagram, for simplicity I will choose the latter approach. Also once you consider the GP state in more detail you realise that several states are missing in the sequence diagram such as interrogating the appointments to see who is next and if they are in the waiting room, requesting the client to come to the consulting room, etc.

Now lets start to use magicdraw.

5. Creating a UML State Diagram

I am working with the PCC project we created in the earlier tutorials.

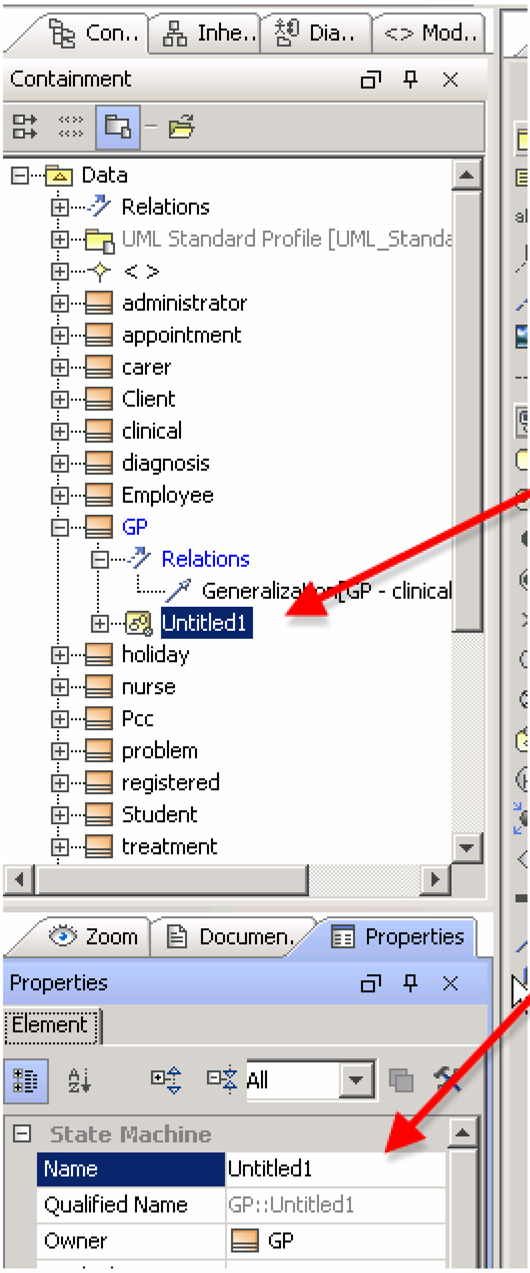


To create a new State diagram that is related to a particular class,

Select the appropriate class in the class diagram,

right mouse click and select:

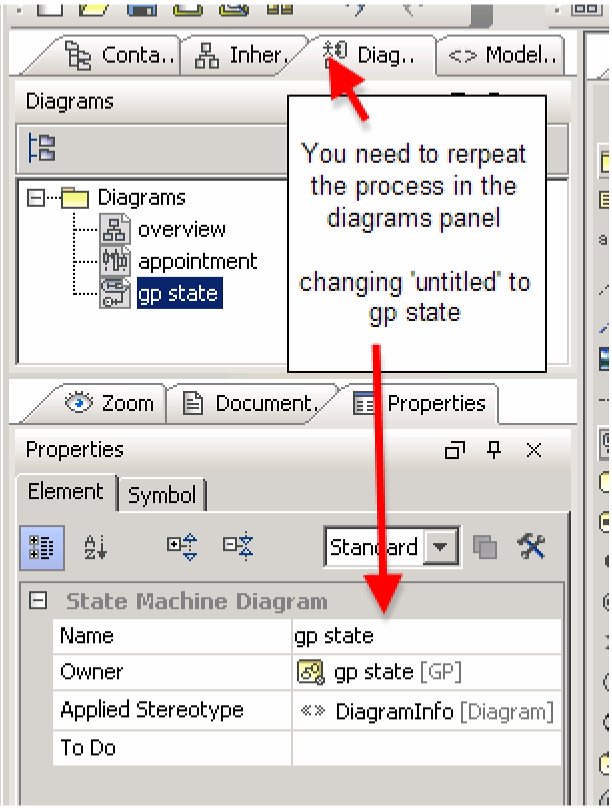
New diagram, State machine diagram



Name	Untitled1
Qualified Name	GP::Untitled1
Owner	GP

In the containment panel you need now to select under data-> GP the 'Untitled1' entry.

Then in the properties panel below it type in "GP state"



You need to rerpeat the process in the diagrams panel

changing 'untitled' to gp state

Name	gp state
Owner	gp state [GP]
Applied Stereotype	«» DiagramInfo [Diagram]
To Do	

Exercise 2

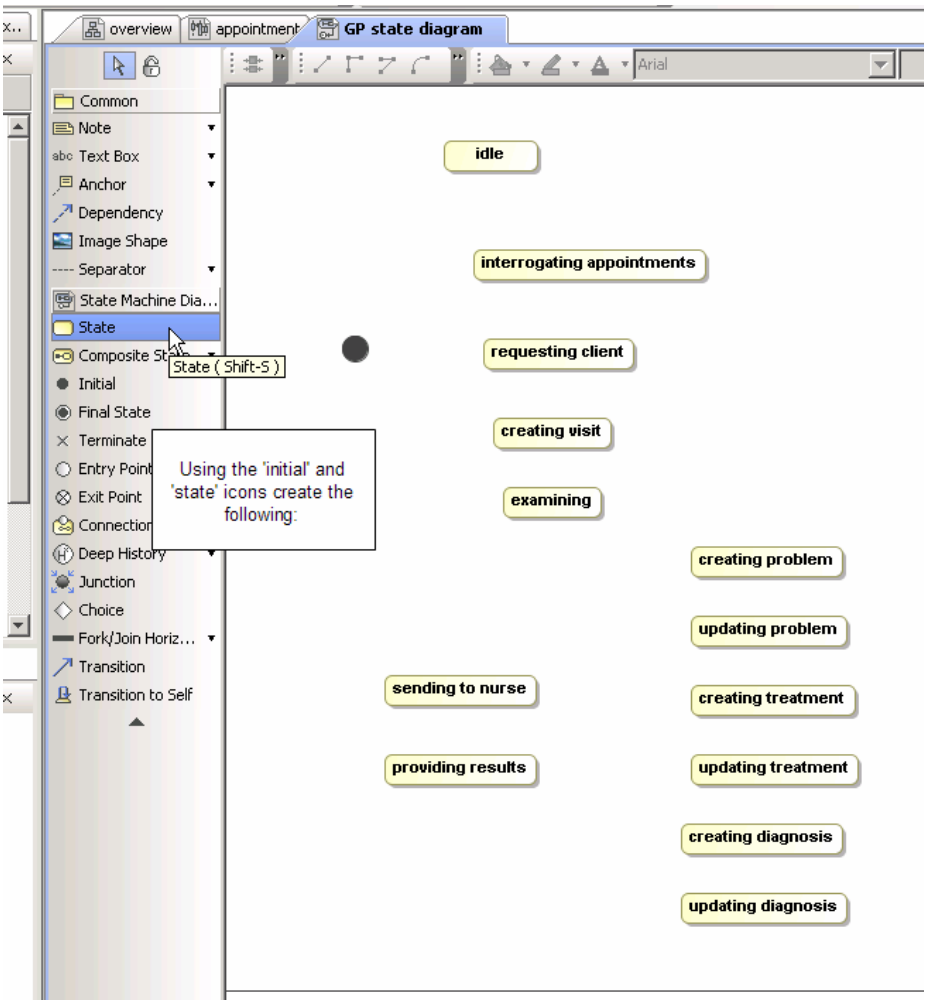
Complete the steps described above.

1.4. Drawing the State Diagram

From our previous discussion, we note that the states that **GP** can be in are:

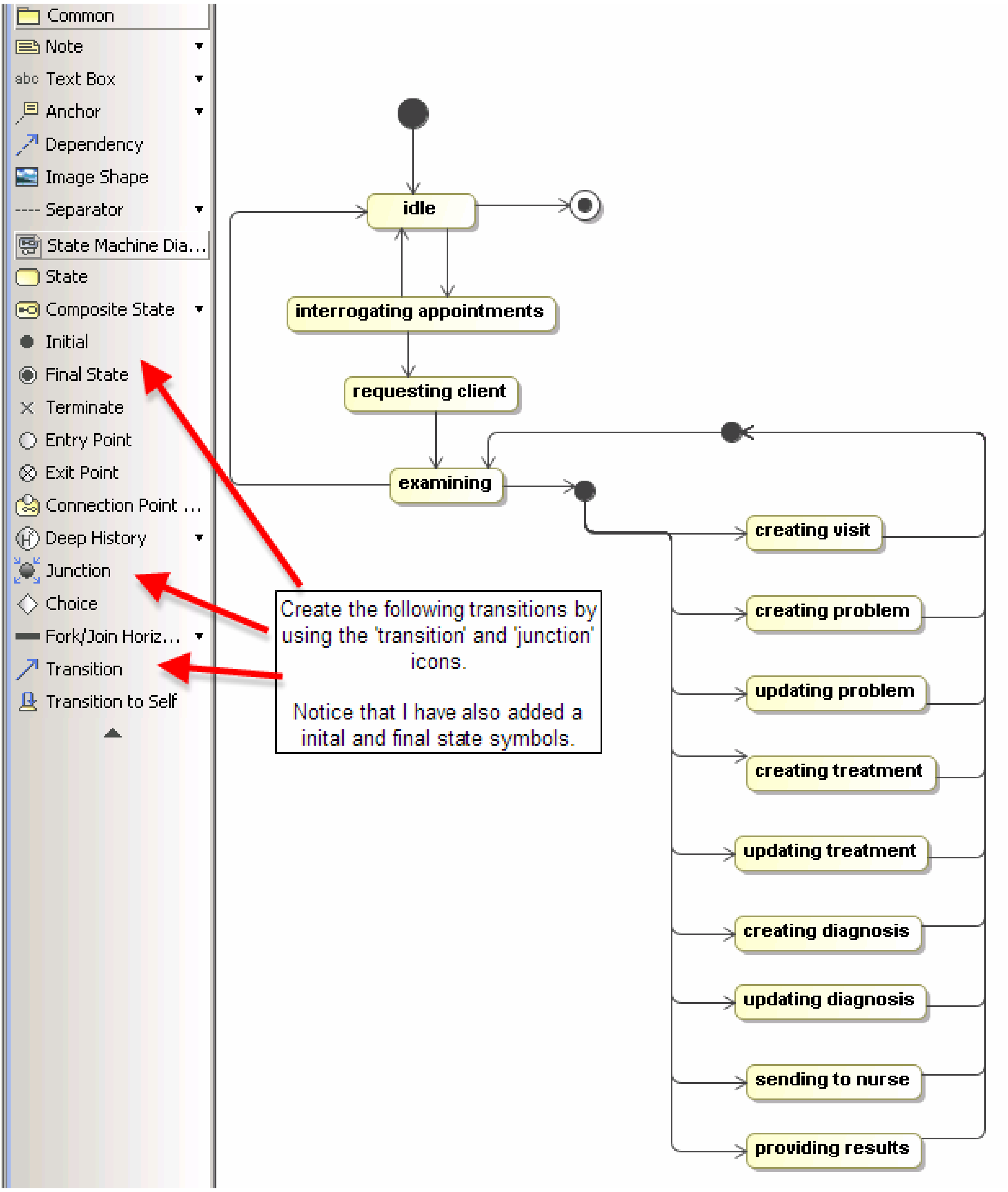
- Idle
- Interrogating appointments
- Request Client to consulting room
- Creating Visit
- Examining
- Creating Problem
- Updating Problem
- Creating Treatment
- Updating Treatment
- Creating Diagnosis
- Updating Diagnosis
- Sending to Nurse
- Providing Results

In addition, every state diagram normally has a start state, which in UML notation is a filled in circle, or large bullet. Let's model this:



Using the 'initial' and 'state' icons create the following:

1.5. Adding Transitions



Create the following transitions by using the 'transition' and 'junction' icons.

Notice that I have also added a inital and final state symbols.

There are two alternatives to the above interpretation:

- The Examining state could have been classed as a superstate with *creating visit* etc. as substates.
- The Examining state could have been classed as a state with *creating visit* etc. as 'do' activities within it.

We could have even modelled the above using a mixture of the above three approaches by considering entry and exit activities for the examining state described next.

Exercise 3

Create the State diagram displayed above.

1.5.1. Naming transitions (adding events)

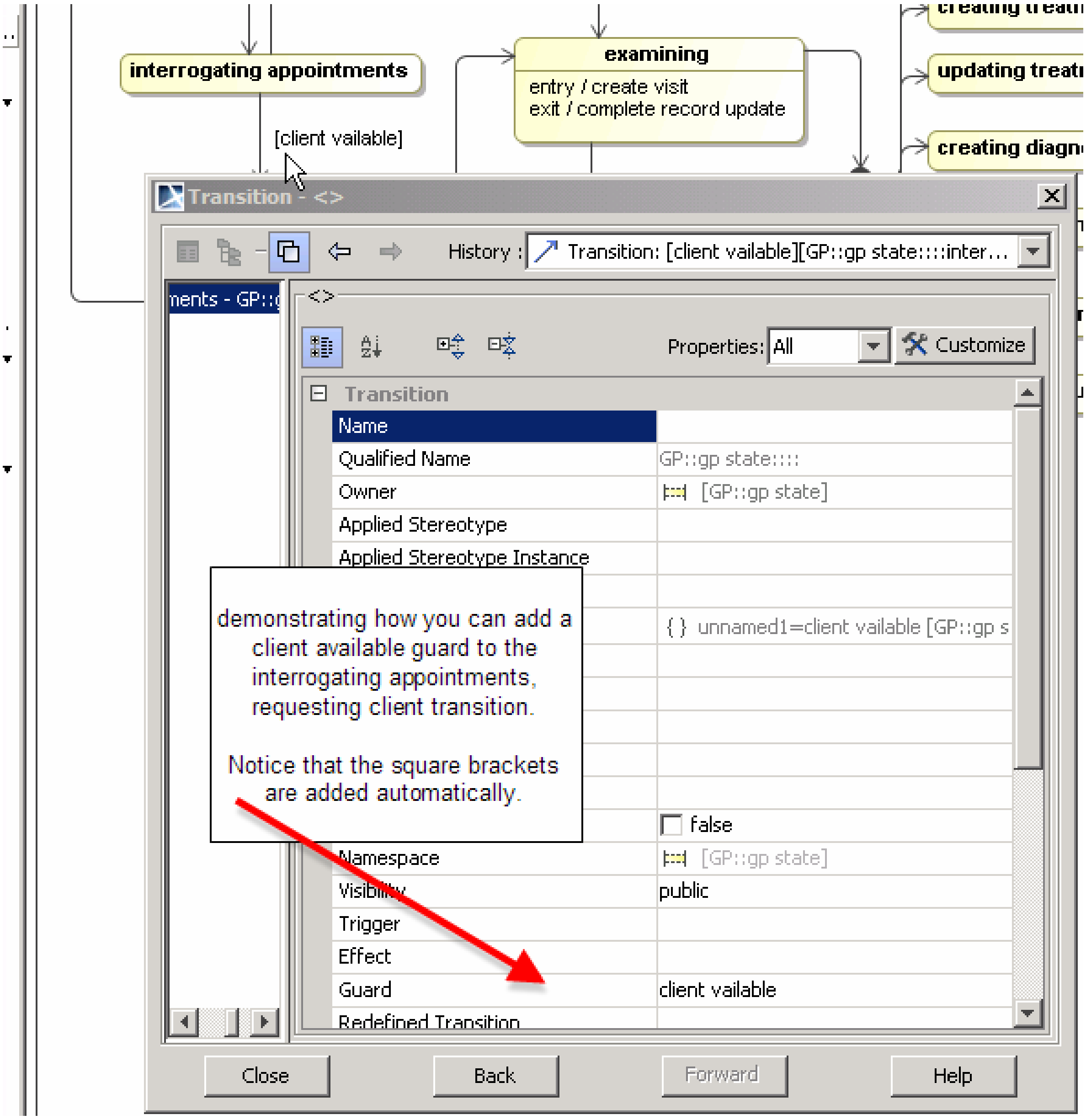
It is easy to name transitions simply by using the specification box for the transition. As mentioned earlier the name may sometimes be the equivalent to an incoming message, and if it is a call message in the sequence diagram a method of the target class.

See section 8.3 in the dynamic modelling knowledge tutorial which describes the important difference between named (i.e. have events) and unnamed (triggerless) transitions.

In draft state diagrams I often do not place events (names) on the transition lines, but this must be done in the final versions of State diagrams if you do not want the transitions to be triggerless.

1.5.2. Guard conditions

You can also specify guard conditions, which have syntax the same as pre uml2.0 see the subsection 6.5.2 single branching in the dynamic modelling knowledge tutorial. The guard in effect means that the transition can only happen when the square bracketed expression is true.



Transition	
Name	
Qualified Name	GP::gp state::::
Owner	[GP::gp state]
Applied Stereotype	
Applied Stereotype Instance	
	{ } unnamed1=client vailable [GP::gp s
	□ false
Namespace	[GP::gp state]
Visibility	public
Trigger	
Effect	
Guard	client vailable
Redefined Transition	

demonstrating how you can add a client available guard to the interrogating appointments, requesting client transition.

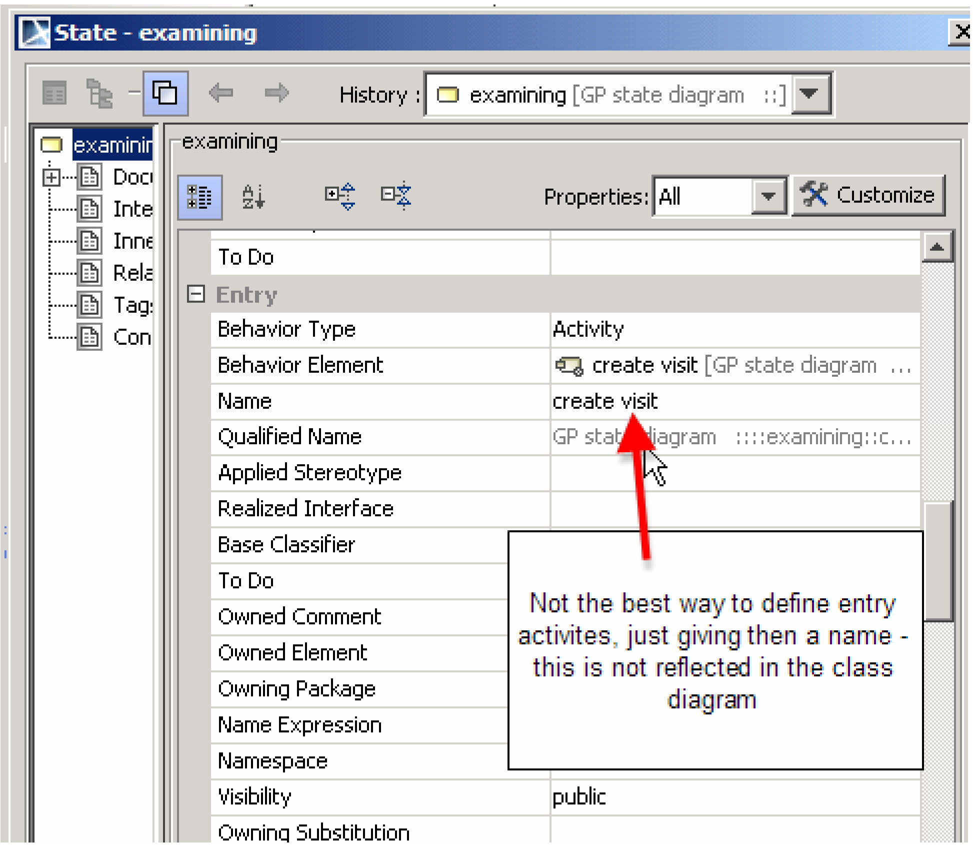
Notice that the square brackets are added automatically.

1.6. Defining Entry and Exit activities

Activities within States frequently are equivalent to methods within the Class and Case tools allow you to model this, unfortunately MagicDraw is rather tortuous in this respect! Obviously there is not always a one to one correspondence of ‘do’ activities to class methods but I find it does help to keep the ratio near that.

1.6.1. Defining activities that do not relate directly to a method - to be avoided

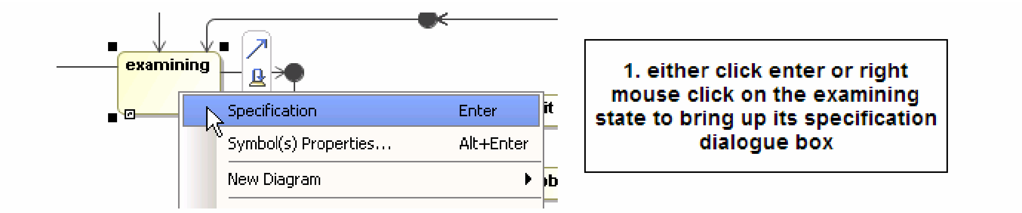
You simply give the activity a name when in the state specification dialogue box for the particular state you have selected. This results in the ‘do’ activity entry appearing in the state in the state diagram but nothing happens in the associated class diagram, as you have not defined it as a method of the class.



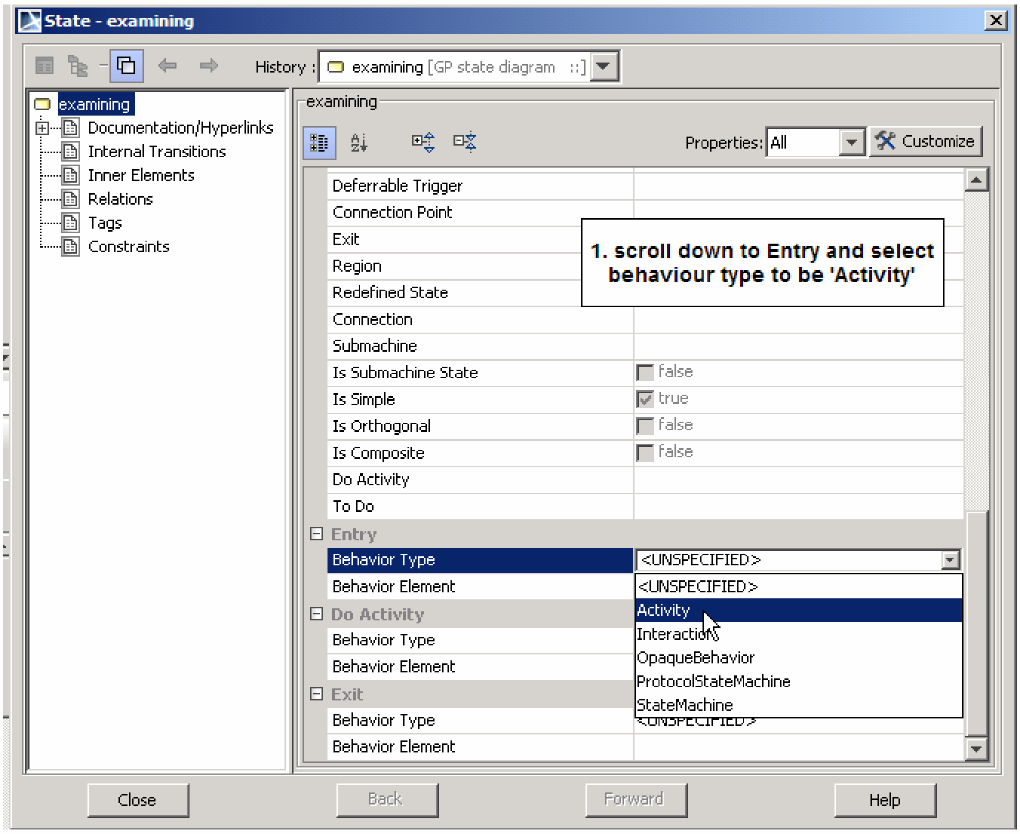
To Do	
Entry	
Behavior Type	Activity
Behavior Element	create visit [GP state diagram ...
Name	create visit
Qualified Name	GP state diagram :::examining::c...
Applied Stereotype	
Realized Interface	
Base Classifier	
To Do	
Owned Comment	
Owned Element	
Owning Package	
Name Expression	
Namespace	
Visibility	public
Owning Substitution	

Not the best way to define entry activites, just giving then a name - this is not reflected in the class diagram

1.6.2. Defining activities that relate (in)directly to a method

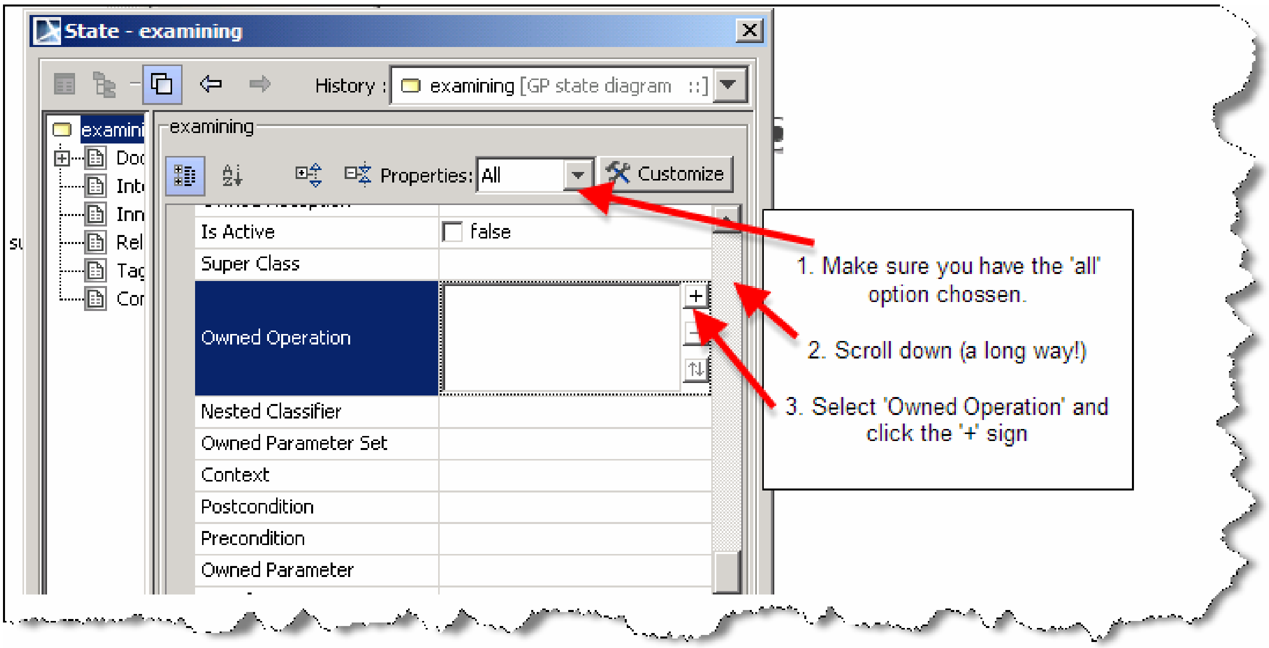


1. either click enter or right mouse click on the examining state to bring up its specification dialogue box



Deferrable Trigger	
Connection Point	
Exit	
Region	
Redefined State	
Connection	
Submachine	
Is Submachine State	□ false
Is Simple	☑ true
Is Orthogonal	□ false
Is Composite	□ false
Do Activity	
To Do	
Entry	
Behavior Type	<UNSPECIFIED>
Behavior Element	
Do Activity	
Behavior Type	
Behavior Element	
Exit	
Behavior Type	<UNSPECIFIED>
Behavior Element	

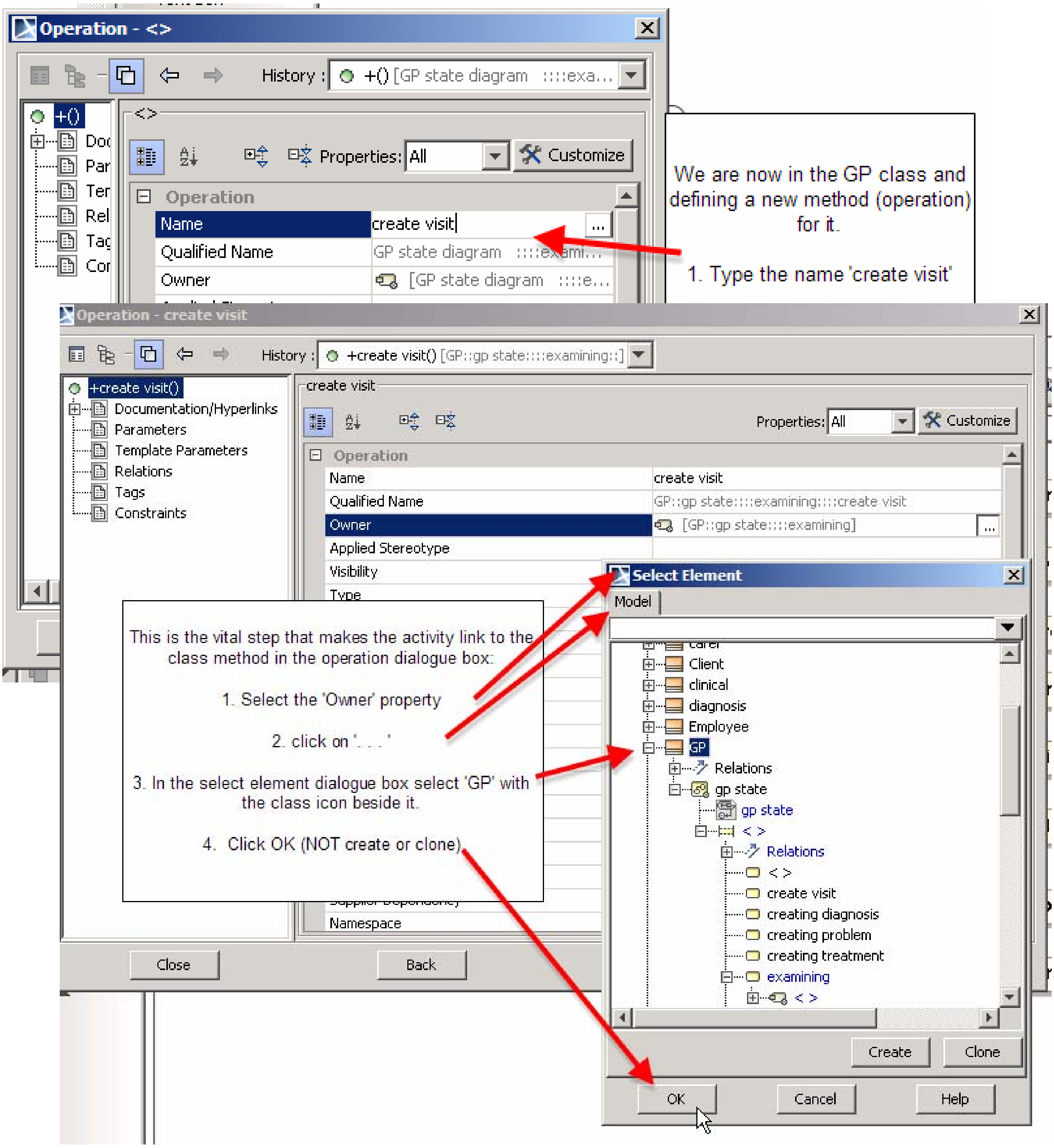
1. scroll down to Entry and select behaviour type to be 'Activity'



1. Make sure you have the 'all' option chossen.

2. Scroll down (a long way!)

3. Select 'Owned Operation' and click the '+' sign



Operation - < >

History : +() [GP state diagram ::::exa...

Properties: All

Operation

Name	create visit
Qualified Name	GP state diagram ::::exami...
Owner	[GP state diagram ::::e...

We are now in the GP class and defining a new method (operation) for it.

1. Type the name 'create visit'

Operation - create visit

History : +create visit() [GP::gp state::::examining::]

create visit

Properties: All

Operation

Name	create visit
Qualified Name	GP::gp state::::examining::::create visit
Owner	[GP::gp state::::examining]
Applied Stereotype	
Visibility	
Type	
Namespace	

This is the vital step that makes the activity link to the class method in the operation dialogue box:

1. Select the 'Owner' property

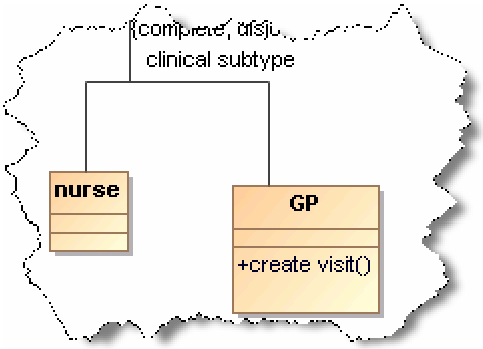
2. click on '. . . '

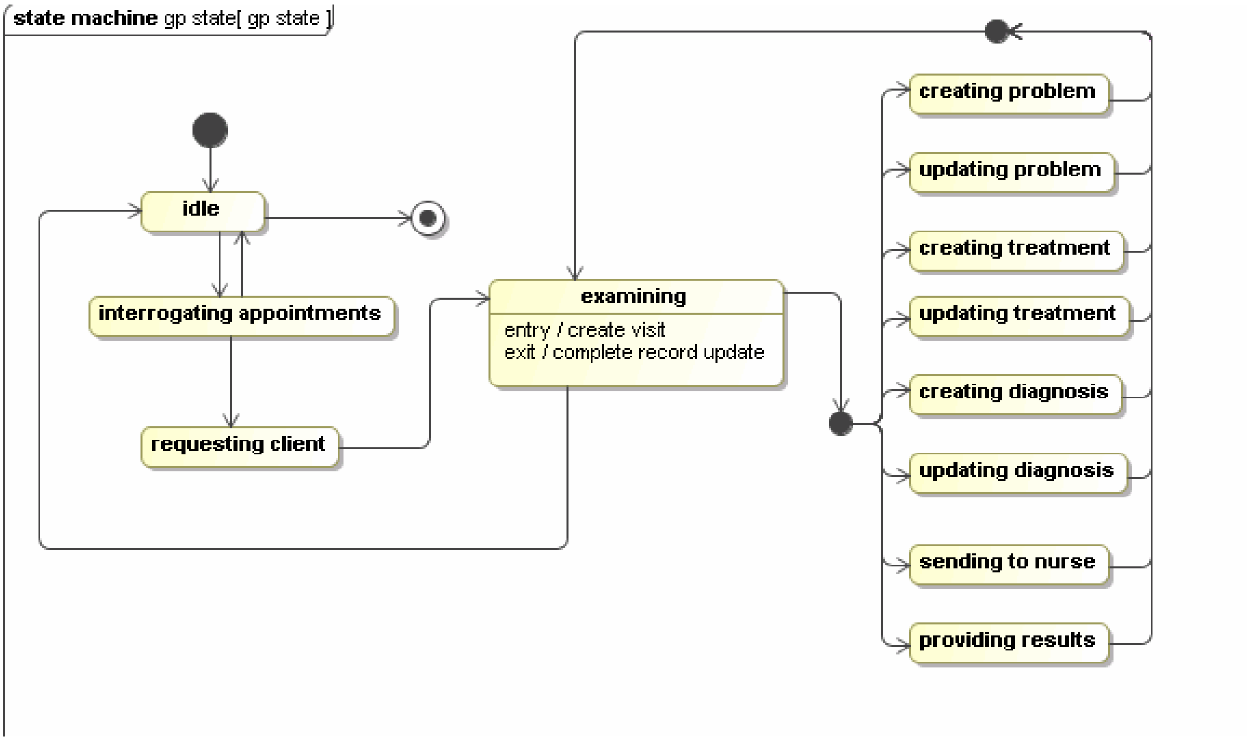
3. In the select element dialogue box select 'GP' with the class icon beside it.

4. Click OK (NOT create or clone).

Looking at the class we now see a new method in the GP class, unfortunately the state does not show the method name and we need to type it in as suggested in the previous approach.

Ambler 2004 encourages, like me, the binding together of class methods and transitions/activities, it is a shame the case tools make this so difficult!





Exercise 4

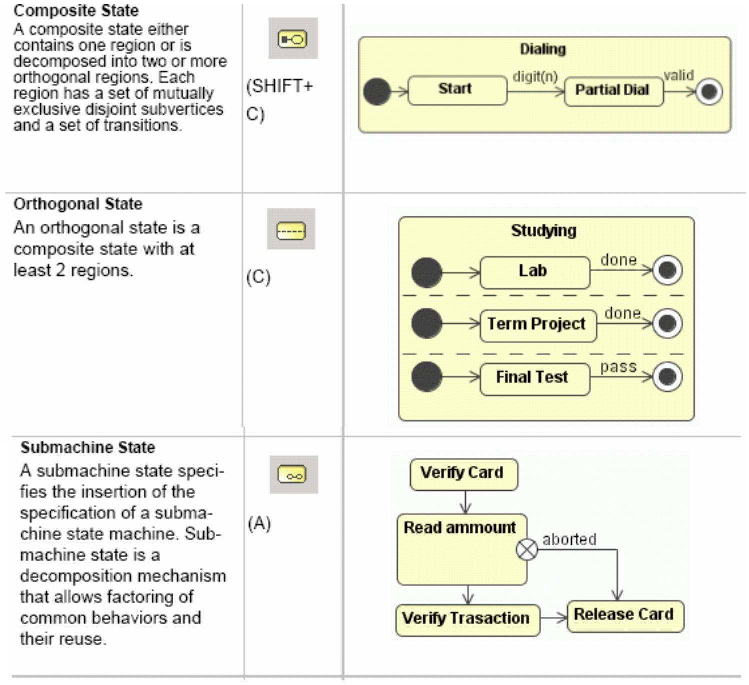
Annotate your diagram to imitate the one shown above, if you have added the guard conditions just leave them in.

1.7. Do Activities

You can also add ‘do activities to the state in between the entry and exit ones. But I’m sure you have got the general idea now so will leave you to experiment more with state diagrams.

1.8. Composite, Substates and Concurrent states

As discussed in the dynamic modelling knowledge tutorial it is possible in uml to model composite states, substates and concurrent states and you can do all this in Magicdraw. The rather sketchy details of the three types are provided below in Magicdraws user manual.

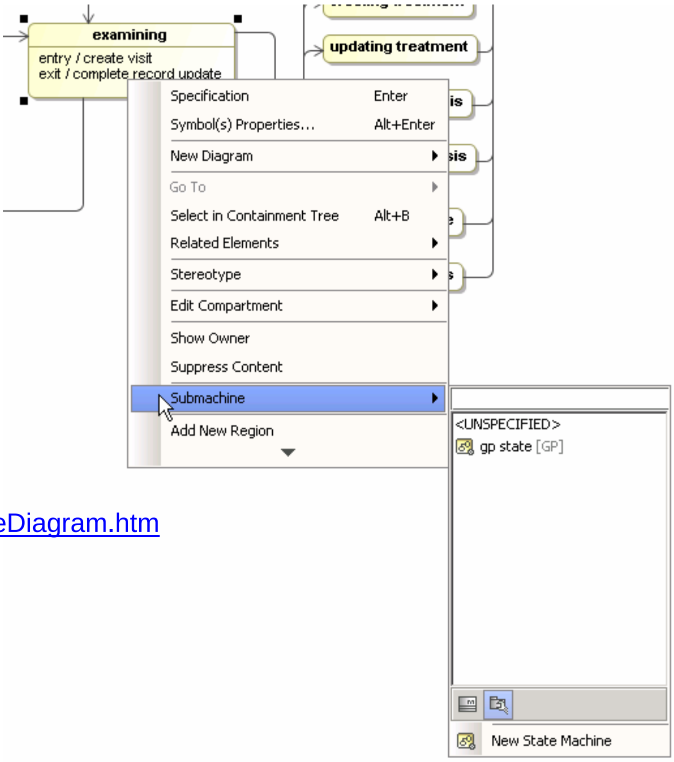


Composite State A composite state either contains one region or is decomposed into two or more orthogonal regions. Each region has a set of mutually exclusive disjoint subvertices and a set of transitions.	(SHIFT+ C)	Dialing: Start →digit(n)→ Partial Dial →valid→
Orthogonal State An orthogonal state is a composite state with at least 2 regions.	(C)	Studying: Lab →done→; Term Project →done→; Final Test →pass→
Submachine State A submachine state specifies the insertion of the specification of a submachine state machine. Submachine state is a decomposition mechanism that allows factoring of common behaviors and their reuse.	(A)	Verify Card → Read ammount →aborted→ Release Card; Read ammount → Verify Trasaction → Release Card

For example suppose you wanted to specify a number of substates for examining, because you felt that all the states you had defined for problem, treatment, diagnosis were really part of examining. To do this you would select the examining state, then right mouse click on it to bring up the menu and select submachine, you could then either select from a list of already defined state diagrams or create a new one.

Alternatively you may have decided to model this situation as a composite state, it depends on which is the correct interpretation.

For a good example of the use of a substate diagram see:



http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

6. Final points

This final tutorial has introduced you to some aspects of State diagramming. These diagrams will probably represent the most detailed level of analysis you will consider in the course, if you went any further you would be investigating a programming language!

You have seen how to associate a state diagram with a particular class and also how to associate an event with a classes operation, important in terms of gaining consistency between elements within the uml model.

Finally I introduced you to some of the more exoteric aspects of state diagrams such as composite and concurrent states as well as substates. While I did not provide you with a detailed example of their use you will find them frequently used in books on UML, particular Blaha and Rumbaugh 2005.

Remember UML is a practical tool – you can only learn to model with UML by applying it to problems and constantly practicing, the scenarios document provides over a dozen appropriate narrative descriptions you can make use of to gain the necessary skills I always start by sketching out a very simple solution on paper, you will be supprosed how practice makes you think of solutions quickly and effectively using the UML framework.

All the best Robin Beaumont August 2007

7. References

Web based example, along with a reference to a good book on UML:

http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

Some basic tips on State diagrams:

http://www.ertin.com/pr_state_diagrams.html

Good example of a state diagram from the book uml 2 for dummies:

http://www.dummies.com/WileyCDA/DummiesArticle/id-2079,subcat-PROGRAMMING.html

For those of you who may want to know how this is converted to a programming language see: http://www.objectmentor.com/resources/articles/umlfsm.pdf

One of Harel's excellent clear papers about statecharts:

http://www.wisdom.weizmann.ac.il/~dharel/SCANNED.PAPERS/Statecharts.pdf

Visual paradigms explanation of State diagrams with references to the superstructure specification 2.0

http://www.visual-paradigm.com/VPGallery/diagrams/State.html

To learn more about transitions / events see UML superstructure specification 2.1.1 page 566 section, 15.3.14 Transition (from BehaviorStateMachines).

Book

Scott W. Ambler 2004 The Object Primer: Agile Model-Driven Development with UML 2.0: Agile Model-driven Development with UML 2.0 (Paperback) £19 see http://www.agilemodeling.com/artifacts/stateMachineDiagram.htm

Acronymophobia
Noun. Condition caused by exposure to overwhelming number of technology acronyms and buzzwords.

End of document 11/10/2007 10:18:24