UML Sequence & Communication Diagrams
in Magicdraw
Robin Beaumont Wednesday, 05 October 2011

Contents

1 Introduction .................................................................................................................. 2
  1.1 Where to obtain the software .................................................................................. 2
  1.2 What do I need to know before I begin this tutorial? ............................................ 2
  1.3 What are the aims of this practical chapter? ........................................................ 2
  1.4 Relationship between Sequence diagrams and Class diagrams .......................... 2
2 The Narrative .............................................................................................................. 3
3 Developing an instance scenario suitable for sequence diagram modelling .............. 4
  3.1 Problems with the terms Narrative & Scenario .................................................... 4
4 Creating a new Sequence Diagram ........................................................................... 5
5 Creating a named instance ......................................................................................... 6
6 Messages ................................................................................................................... 8
  6.1 Signal messages ..................................................................................................... 8
  6.2 Create messages .................................................................................................... 9
7 Creating a call message .............................................................................................. 11
8 Creating operations in classes directly ...................................................................... 12
9 Using operations already in classes to define call messages ..................................... 13
10 Adding parameters and arguments to messages ...................................................... 14
11 Changing the appearance of the diagram ............................................................... 15
12 Call messages naming different from operation evoked ......................................... 16
13 Interaction uses ...................................................................................................... 17
14 Loops and alternative Occurrence Fragments ....................................................... 18
  14.1 Pseudo code ....................................................................................................... 19
15 Synchronizing Diagrams ......................................................................................... 20
  15.1 Why no Create operations? ................................................................................. 20
16 Communication Diagrams ...................................................................................... 21
17 Revision - Magicdraw message types .................................................................... 23
18 Web links ................................................................................................................ 23
1 Introduction

1.1 Where to obtain the software
This depends on who you are:

- Students at Bath or The Royal College of Surgeons (Edin.) will be provided with the software along with the academic licence.

1.2 What do I need to know before I begin this tutorial?
Before you work through this tutorial you should have the following knowledge and skills:

Experience of using Magicdraw having worked through the following three practical tutorials:

- Introduction to ERD modelling using UML Class diagrams with Magicdraw
- UML Class diagrams with MagicDraw
- UML Use Cases using MagicDraw

All the above are available from [http://www.robin-beaumont.co.uk/virtualclassroom/contents.html](http://www.robin-beaumont.co.uk/virtualclassroom/contents.html).

Have a working knowledge of Entity Relationship diagrams, UML Class, sequence and use case diagrams, preferably having worked through:

- Introduction to Uml part 1 - Class/instance modelling using UML
- Introduction to Uml part 2 - Associations
- An introduction to dynamic modelling and process re-engineering using UML - part 1

All the above are available at [http://www.robin-beaumont.co.uk/virtualclassroom/contents.html](http://www.robin-beaumont.co.uk/virtualclassroom/contents.html).

1.3 What are the aims of this practical chapter?
This practical chapter is the fourth in a series to introduce you to using a specific CASE tool, MagicDraw Personal Edition (MD/PA). This tutorial assumes that you have worked through the first three tutorials so in several places it rather briefly describes what to do. By the end of this chapter you will feel confident about using MD/PA to draw UML compliant Sequence diagrams. The example we will work through will make use of the narrative on the following page.

1.4 Relationship between Sequence diagrams and Class diagrams
This will become clear as you progress through this chapter. Two basic rules:

A sequence diagram usually represents instances from classes in the class diagram. Consequently:

- the sequence diagram provides details about a subset of the classes identified.
- A class instance can be represented by zero (i.e. no patient instance), one or many times (i.e john Brown and Jane Tyler) in a sequence diagram

Certain messages equate to a operation of a particular class (the operation being in the class where the arrow head is).
2 The Narrative

A Primary care centre (PCC) consists of Employees, Clients (patients), Voluntary workers and Students. In terms of how the employees are paid they are classified as being either casual, part-time, full time or honorary staff. General Practitioners, a particular group of employees, can be either qualified or in training which because both are salaried are different from the various other students that are at the PCC. Nurses (of which there are several varieties) carry out consultations, home or institutional visits, client teaching sessions (one to one and group) and various clinics such as toddler, ulcer and diabetic management.

Patients can either be registered or visitors, either can see a variety of the above people. They may see a person for either an individual or a planned series of visits. A visit may be a group session/consultation or one or more treatments (blood and/or urine test or just Blood pressure check, etc).

Each client's record has several aspects. One aspect consists of one or more Problems which may be open, referred, being managed or resolved. For example a patient presenting with a leg ulcer may be referred by the GP to a practice nurse who will dress the wound until it is healed, The GP may request a follow up appointment which may be either a one off event or several (such as every two weeks for 10 weeks). Associated with each problem may be specific treatments (each which may relate to more than one problem) However the treatment will always only relate to a single patient. Alternatively the GP may just refer the client to a consultant, where the Problem would have the status ‘referred’.

Another aspect of the client's record are the diagnoses. Clients may have zero or more diagnoses which may be linked to a particular problem and/or specific treatment sometimes a diagnosis may be a stand alone detail such as Klippel-Feil syndrome.

The client record also contains appointment details which may be either (missed, attended, patient abandoned or practice abandoned), once the client actually sees the person (usually the person they have the appointment with) a visit is recorded. The visit can be with anyone discussed above, a visit may be with more than one person, such as a GP and a trainee GP.

The PCC makes use of both the BFI for advice about various treatments as well as an in house formulary both of these are available electronically.

Another group of people) are the administrators (who can be employees, voluntary workers or students) such people operate various phone and reception services offered at the PCC. They vary from operating the front desk (logging and possibly editing appointments), to arranging repeat prescriptions and organizing telephone consultations with GPs or nurses.

The voluntary workers are managed by a voluntary worker co-coordinator who is also herself a voluntary worker.

In a previous tutorial “Class diagrams in UML” we developed the above Class diagram using several aspects from the above scenario. The above scenario describes the situation in terms of Classes rather than instances, which is fine for class modelling, however for sequence diagramming we need to think in terms of instances, in other words we need to be able to convert the above into a series of interactions of specific instances, which we will now do.
3 Developing an instance scenario suitable for sequence diagram modelling

The above narrative is very much an overview and from it we could develop a large number of specific interactions for the various classes. Blaha & Rumbaugh 2005, within the context of use case modelling, divide specific scenarios (i.e. ‘instance scenarios’) into 5 types:

1. Normal behaviour
2. Variations in normal behaviour
3. Exception conditions
4. Error conditions
5. Cancellations of requests

While I do think it is not too important to understand the exact differences between each of the above proposed categories I think it is valuable to appreciate the wide range of behaviours that should be expected.

I have produced a possible normal behaviour instance scenario below:

1. Joan Smith (client) phones Ann Page (administrator) to make an appointment with Dr Coates (GP).
2. The administrator provides a suitable time and creates an entry in the diary (give it a unique instance 23456).
3. At the specified time Joan Smith turns up and introduces herself to Steve Brown (administrator) who updates the appointment 23456, setting the status to arrived.
4. Dr Coates (GP) checks the appointment 23456 to see if the patient has arrived, discovering she has then requests her to make her way to the consultation room.
5. Joan Smith presents herself at which point Dr Coates creates a visit (give it a unique instance 1789) record.
6. Joan Smith has a long history of hypertension and Dr Coates updates her hypertension problem record (number 35463) after checking and recording her Blood Pressure (unique treatment instance 1234).
7. but also she has a new problem, oral Thrush (create problem record 6789) for which Dr Coates proscribes Nystatin pastilles (creates unique treatment instance 1235).
8. Dr Coates adds oral candidiasis to Joan’s list of diagnoses (creates a new instance of diagnosis unique number 143).
9. Dr Coates then provides Joan with both feedback about the consultation and instructs Joan to make an appointment for two weeks time when leaving.
10. Joan arranges the appointment with Sarah Jane (instance of administrator) who provides her with the details and creates an entry in the diary (giving it a unique instance 23678).

Taking information from this instance scenario we will now consider some dynamic aspects of it to demonstrate various aspects of UML Sequence diagramming in Magicdraw.

3.1 Problems with the terms Narrative & Scenario

While I have used the terms Narrative and Scenario above to try and represent different things in my mind; A narrative being a high level general description, not usually describing instances and a scenario being a detailed description of a sequence of events that happen to a specific number of instances. This is largely my own creation. So be warned the two terms are frequently used interchangeably. In fact the document scenarios for modelling I have created should more correctly be called narratives for modelling if I follow my own descriptions!
It is assumed that you have already created the PCC project and have it open.

Your list of classes will be longer than the one above if you completed (you should have done!) the extended exercise in my Magicdraw Class diagrams tutorial.
5 Creating a named instance

There are several ways to create a named instance in a sequence diagram. I will mention a rather complex way first of all as it demonstrates the use of the containment tree which is often ignored but provides a useful overview of what you have in your project at any one time.

The screen shot opposite shows the containment tree before we have added anything to the sequence diagram.

We first need to add a un-named instance of a class before creating a named one. A quick way to do this is to select one of the classes in the containment tree and then drag it across to the drawing canvass, in the example below I have selected the client class.

Once you have done this, expanding the containment tree for the appointment branch by clicking on the ‘+’ shows that we have two new elements in the tree, shown opposite. The icons beside each indicating that one is a property and one is a lifeline. Double left mouse clicking on the client property allows us to give it a name in effect creating a named instance of the client class. I have given the instance the name smith in the screenshot below.

Quick shortcut: instead of the above method you can simply double click on the name box at the top of the lifeline and then type in the instance name – it achieves the same thing.
Exercise

You need to follow the instructions on the previous page to create the required instances to model the instance scenario described, here are mine.

Things to note from the above exercise:

- While you can move the various instances of the classes you have created along the top of the diagram (horizontally) you cannot move them vertically.
- You can have several instances of one class.
- You can have classes that are not based upon people - remember a class is anything you wish to collect information about.

Add any other points you have discovered while completing the above task:
6 Messages

It is important that you revise the section on messages in 'An introduction to dynamic modelling and process re-engineering using UML - part 1' available at [http://www.robin-beaumont.co.uk/virtualclassroom/contents.html](http://www.robin-beaumont.co.uk/virtualclassroom/contents.html).

We will create each or the main types of message starting with a simple signal message.

6.1 Signal messages

Suppose we want to create a request_appointment message from smith to ann_page:

Simply select the send message (asynchronous signal message) icon and select the smith:client lifeline then drag (or click) to the ann_page:administrator lifeline.

To give it a name simply type in the dotted box.
6.2 Create messages

You can also create a message that specifies that the message results in the creation of a new instance, for example when Ann Page creates appointment 23456.

Simply select the appropriate message icon, in this instance Create Message, then click on the originator lifeline (i.e. ann_page) and then drag to the target one (i.e. 23456:appointment). Finally add the name for the message, I have called it create.

Exercise

Create the two messages described previously.
Exercise

Below is a first attempt at creating the full instance scenario – try to reproduce it yourself, at the same time noting any major errors/omissions I have made.
7 Creating a call message
Suppose we want to create a message that evokes a request_appointment operation in the administrator class: In other words the request_appointment message once it reaches the administrator class, causes it to run the request_appointment operation.

This time we just select the appropriate message sort, the call message (=synchronous call message), having first deleted the previous message.

Draw the message line between the two instances as we have done before.

By clicking on the bubble that appears on the line we are asked to name the operation that the message will evoke. I have called the operation request_appointment.

The result is that a new operation is also created in the Administrator class and if we check the class diagram we see it has been updated:

We can also do this the other way around create a call message that evokes a operation that is already defined for a class.

Exercise
Create the message described above.
8 Creating operations in classes directly

There are many ways to do this, you can select the class in the containment tree then right click and select new element and then operation. or alternatively in a class diagram where you have the class you wish to create the operation for select the class then click on the insert new operation bubble.

A third method, making use again of the containment tree is to simply select the class but this time double left mouse click on it, this brings up the specification dialog box for the class, by opening up the operation tree on the left hand side you can create a new operation.
Exercise

Using any of the approaches described on the previous page create new operations for the administrator and appointment classes so that you end up with:

```plaintext
+request_appointment()
+check_appointment()
+setAvailability()
+update_appointment()
```

Hint: check the containment tree after you have finished to see that you have what you think.

9 Using operations already in classes to define call messages

In our scenario we have:

At the specified time Joan Smith turns up and introduces herself to Steve Brown (administrator) who updates the appointment 23456, setting the status to arrived.

We already have in the appointment class the update operation so all we need to do to model this situation is to draw a call message in our sequence diagram evoking this operation.

we do this by first drawing either a signal or call message between the two relevant instances (steve_brown_administrator to 23456:appointment) then select the line NOT the bubble, then right mouse click on it to bring up the following popup menu, moving down to either of the call message types gives us a list of the methods in the target class. In this instance select the asynchronous call message and then the update operation.
10 Adding parameters and arguments to messages

In the previous section we ignored a subtlety:

At the specified time Joan Smith turns up and introduces herself to Steve Brown (administrator) who updates the appointment 23456, setting the status to arrived.

Although we send a message we did not say exactly what was in it you can do this by specifying parameters.

I have now added an attribute to the appointment class called status, what we need is for the message described above to update the value for the status attribute in our instance to arrived. To achieve this we need to add a parameter value to the update() operation. So we can say update (status=arrived).

To do this we need to edit the specification of the update operation, using the containment panel selecting the update operation of the administrator class then double left mouse clicking on it the specification dialog box appears. selecting the Parameters option in the left hand tree we now have the opportunity to create parameters for the operation.

We can now specify the parameter (see screenshot below). I have created a parameter called status with a default value of 'not arrived'.

This now updates the message to:

Simply type in after the equals sign (=) arrived so we end up with:

This shows that the message called the update operation in the appointment instance which updates the status value to arrived.
11 Changing the appearance of the diagram

While you were creating the diagram you will have noticed several, possibly irritating points, the lifeline tops are fixed and there appears no way for reducing the lifeline lengths, or the activation bars beyond a certain length. However having said all that there are several options you can choose from to change the appearance of the sequence diagram. Look at the details below.

Exercise
Show message numbers
Show activations
Show diagram info
Show diagram icon
12 Call messages naming different from operation evoked

Unfortunately in uml 2 (Bennett, Skelton & Lunn 2005 page 165) if you have a message that is a call message it takes the name of the operator so that it is no longer possible to display both a message name and the operation it invokes. This has been implemented in both MagicDraw and Visual Paradigm in contrast to the AND situation which was possible in versions of System Architect.

If you do not use a “call message” there is no specific way of linking a message with a operation. Things seem to have gone backwards!

A warning, once you have created a call message and the associated operation simply deleting the graphical object does not delete the actual operation you have created, to achieve that you need to find it in the containment pane and delete it from there.

The picture below shows what I miss so much from System Architect.
13 Interaction uses

Interaction uses allow you to link various sequence diagrams so that no single one becomes unwieldy which is always very useful. For instance, even in our simple appointment example the sequence diagram is beginning to look rather cluttered, and interestingly if you look at the appointment making part of it the same messages seem to be repeated, at both the start and end of the scenario. Using a interaction Use can make this diagram clearer and allow us to model the actual appointment arranging process in more detail.

When the Create diagram dialogue box appears we will give this new sequence diagram the name create_appointment.
14 Loops and alternative Occurrence Fragments

I have used the Loop and Alt occurrence fragment drawing icons to create a more detailed explanation of what happens when our client smith books an appointment in the new sequence diagram. I have also annotated the original sequence diagram removing the create appointment message as this is now achieved in the create_appointment sequence diagram (see the picture below).

Exercise

Try to minic, or offer an improved version, of the sequence diagram opposite
Alternatives to showing all the control aspects in sequences diagrams

You may feel that the above sequence diagram is getting overly complex with the loop and branching (alt) structure. UML offers another type of behaviour diagram called an Activity diagram which is a bit like an old fashioned flow chart and some people prefer to use this to show detailed control, particularly branching and looping etc.

You should remember that primarily a Sequence diagram is for a specific sequence of interactions you should not try to create one sequence diagram that shows all possible interactions. As a general rule you will create many more sequence diagrams than class diagrams.

14.1 Pseudo code

This is the mid point between English and a particular programming language. The idea is that it allows people to express the general idea behind what they want to do using the basic semantics of programming without being constrained to a single language. You can learn more about it at http://en.wikipedia.org/wiki/Pseudocode

I have converted the last occurrence fragment we worked onto pseudo code below, I personally prefer pseudo code to showing all the conditional / looping message flows in a sequence diagram, but that is probably simply to do with familiarity and age!

```
If not booked
    While available not true
        Client:smith request_appointment
        Ann_page:administrator checkavailability
        :appointment report back availability status
        If status available Set booked to true
    End While
Else Create appointment
    Confirm appointment with Client:smith
```
15 Synchronizing Diagrams

One of the most important things about a good case tool like MagicDraw is that it keeps all the information from the various diagrams synchronised, for example whenever we added a new call message (i.e. one that calls a method/operation in the target object) anywhere on any class diagram where the class appears the new method/operation should automatically also appear. Looking back at the class diagram we can see this has happened.

15.1 Why no Create operations?

Within the sequences diagram you will remember that we created several ‘create’ messages yet these do not appear as operations in the class diagram, this is because there is a convention within object oriented circles that each class possesses two operations Create and Destroy to allow for the creation and destruction of instances of the class, in other words all classes have the innate ability to create and destroy instances of their own type.
16 Communication Diagrams

The Communication diagram (formally called a collaboration diagram) is an alternative to the Sequence diagram for modelling instance scenarios in your system. Although Sequence diagrams make it easier to see the chronological flow of events, communication diagrams enable you to show links between instances, attribute values, and visibility.

In the past several Case tools provided the functionality to allow you to convert between Sequence and Collaboration diagrams as they were semantically equivalent, unfortunately with the introduction of interaction occurrences in UML 2 the mapping has become more complex and neither MagicDraw or Visual Paradigm offer this functionality. I would suggest to avoid possible duplication and errors creeping into the modelling process, until this functionality is offered within the case tool you should stick to either Sequence or Communication diagrams, in which case I personally prefer sequence diagrams.

To create a new communication diagram:

The screen shots on the following page provide details of how to create the first message, request_appointment, we created in the first sequence diagram before we added the interaction occurrence.
I do not want to dwell too much on communication diagrams as they are really alternatives to Sequence diagrams and people tend to use either one or the other. From this tutorial you can guess which I prefer!
17 Revision - Magicdraw message types

The following provides a resume of the different message types. Remember that Fowler warns that it is dangerous to try to use too many different message types and suggests that you stick to asynchronous messages if you are unsure.

18 Web links

A good introduction to Sequences diagrams and the use of UML 2 semantics by Donald Bell, IT Specialist, IBM can be found at: [http://www.ibm.com/developerworks/rational/library/3101.html](http://www.ibm.com/developerworks/rational/library/3101.html)