Tutorial: your first steps with Narval

LOGILAB S.A.
10, Rue Louis Vicat - 75015 PARIS
Tél: (+33) 1.45.32.03.12
Web: www.logilab.com
Email: contact@logilab.fr

Olivier Cayrol
Alexandre Fayolle

Tutorial: your first steps with Narval

by Olivier Cayrol and Alexandre Fayolle

Prerequisite: To perform the operation described in this section, you should have completed Narval installation. If you had some troubles with this, please send us an email to this address: `contact@logilab.com`, this will help us improve the installation process.

Warning: Narval is an Intelligent Personal Assistant framework that counts a dedicated language, an interpreter, and a graphical user interface that's also an integrated development environment and is called Horn. From now on, we'll use the same term *Narval* for the language, the interpreter or an assistant set up with Narval (a running instance of the interpreter that is executing a set of recipes). But we'll try not to say "to start your Narval, run Narval so that it executes your Narval program" ;-)

1. Starting with Narval

As you probably know, there are two parts in Narval: Horn is the user interface, which you can use to edit recipes or create new ones, and Narval is the program that will run these recipes. Horn can also connect to a running instance of Narval to monitor the interpreter, and start plans remotely.

1.1. Launching the user interface

You can lauch Horn by typing **`horn`** on the command line, to see things and start editing recipes. When prompted, choose the second option (load data from a local directory). You will notice that Horn has created a `.narval` directory in your home directory (or `narval_data` in your application data directory if you're running Windows). You can go and have a look at the contents of this directory. Be careful and do not edit things blindly until you are familiar with the piece of software! Maybe now is a good time to customize some data in `data/memory.xml`, such as the path to your mailbox (unix systems only), and your email address.

1.2. Installing Narval applications

Now, I guess that you are eager to run Narval. You'll have to be a bit more patient: unless you already have written recipes, there's nothing Narval can do for you, since the distribution only includes a set of basic actions and transformations, and no recipes. You shoud therefore download a few applications first, from the Narval application repository [http://www.logilab.org/narval/app.html]. Try *Infopal* or *PDA* for instance.

You can install these application to your narval directory by running the **`npm`** (Narval Package Manager) utility. This can be done directly in Horn, by selecting File->Install package... in the menu, and then selecting the package in the file selector.

1.3. Running recipes

You should open another terminal, and start Narval with the `--socket-manager` and

`--listen-on 8787` command line options. If you are using Windows, using the start menu shortcut will launch Narval with these options. The first option tells Narval that it should be waiting for connections on some ports specified by special elements in its memory and pass these connections to recipes. The second one tells Narval that it should expect monitoring applications to connect on port 8787. We are going to do just this in an instant.

When Narval has finished initializing, connect Horn to Narval using Narval->Control->Connect... in the menu. This will display a connection dialog, with the different fields prefilled to the right values (if you used port 8787). Using Narval->Display->Memory will display the memory window, where you will be able to see what is in Narval's memory. You should also for this quick start enable Narval->Monitor plans, so that plan execution will display a new window in Horn.

2. The PDA recipe

This recipe is part of the PDA (personal digital assistant) application. Make sure you've installed it first.

Once you've launched Narval and accessed the http://hostname:7776/ [http://localhost:7776/] address in your browser, you are on PDA main page.

- **Url Visited:** displays the contents of the file in which Narval writes the URL you recently visited (see Section 3.).
- **Calendar/agenda:** displays a very basic agenda and allows you to enter some meetings or tasks.
- **AddressBook:** displays a very basic address book and allows you to add new entries.
- **Recipes:** displays the recipes available in Narval.
- **Plans:** displays the plans currently run by Narval.

Of course, these are only the very few basic things Narval can do for you. Very soon, thanks to the URL visited and its artificial intelligence, it will be able to guess your interests and to suggest you new URLs; thanks to the agenda, it will remind you your appointments; thanks to the address book, it will be able to automatically answer to the people that send you emails or even phone calls.

3. The `Proxy` recipe

This recipe is part of the Infopal application. Make sure you've installed it first.

This recipe can behave as a web proxy for your browser. If you setup the proxy in Mozilla to be **`hostname:7777`**, Narval will act as a proxy for you. It will log the URLs you visit (and you will be able to browse through it using the PDA recipe). It will be able to filter out banners using a `Junkbuster` module in the next release of Narval.

If you normally use a proxy to browse the web, you should tell Narval the url of this proxy.

This is done by editing the file `~/.narval/data/memory.xml`. In this file, there is a line describing a standard http proxy: <proxy type="http" hostname="proxy" port="3128"/>, you should edit it so that it matches your own configuration.

4. The `Newspaper` recipe

This recipe is part of the infopal application.

This recipe can download various RSS sources on the internet (the one that My Netscape uses) and compile them into an HTML page, that it will put in the `NARVAL_HOME/data/newspaper.html`. It is easy to customize the format of the page to suit your taste, and then use it as your home page (or include it in your home page using server side includes for instance)

To launch it, use Horn: connect to the running instance of Narval, and in the memory window, right click on the news.newspaper recipe and select "instaciate plan" in the popup menu. Depending on your connection and the availability of the various resources, you should get the `NARVAL_HOME/data/newspaper.html` in a couple of minutes.

If you want to add resources, just edit the recipe using Horn. Using Horn to edit recipes is explained in the next chapter.

5. Writing your first recipe

Before you write your first recipe, please check you have the writing rights on the files in `NARVAL_HOME/recipes`.

Now, we will together build a recipe that reads an HTML page on the Web and saves it on your local disk. Thanks to Horn and Narval, this can be done with a few mouse clicks.

1. Launch Horn. When prompted, choose to run with local data.

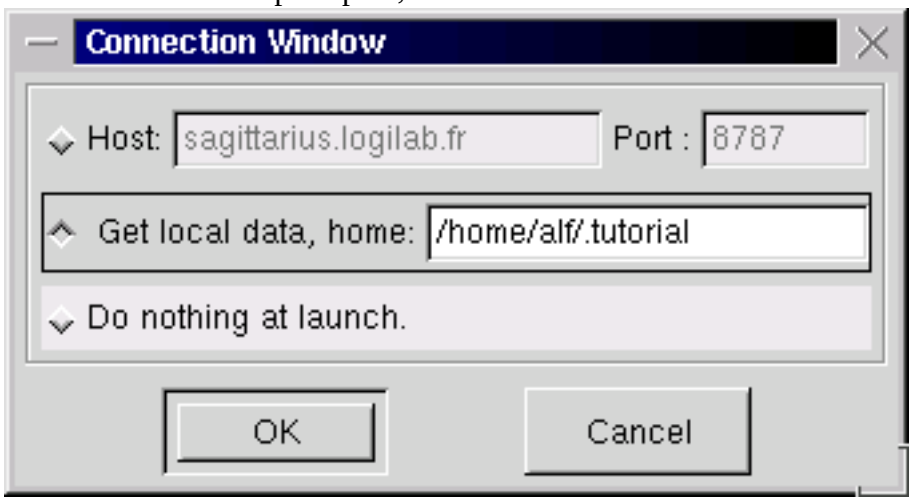

Figure 1. Choosing to get local data

2. Create a new recipe
   a. select Actions->News recipe... in the menu
   b. Choose the cookbook in which your recipe will be added (let's say **`my_recipes`**) and type the name of your new recipe, for instance **`MyFirstRecipe`**

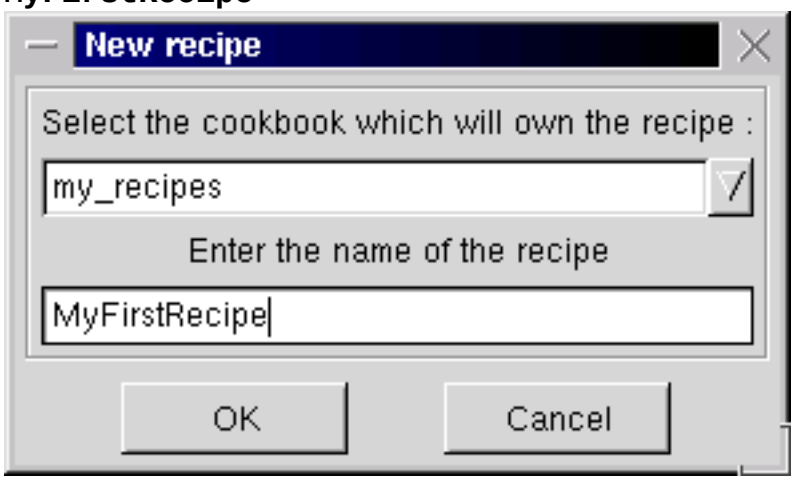

Figure 2. Creation of a new recipe

   c. Check that a new recipe has appeared in the Memory window
3. Edit the new recipe
   a. select the new recipe in the Memory window, right-click on it. This pops up a contextual menu.
   b. select Edit recipe in the contextual menu. Horn opens a blank window called "Edition of recipe MyFirstRecipe".
4. Insert the `Basic.NOP` action. This action does nothing, it is used so that a transition can be inserted before the first useful test, that will check for the required arguments of the step.
   a. select Actions->Add step... in the recipe edition window menu. This brings up a selection window

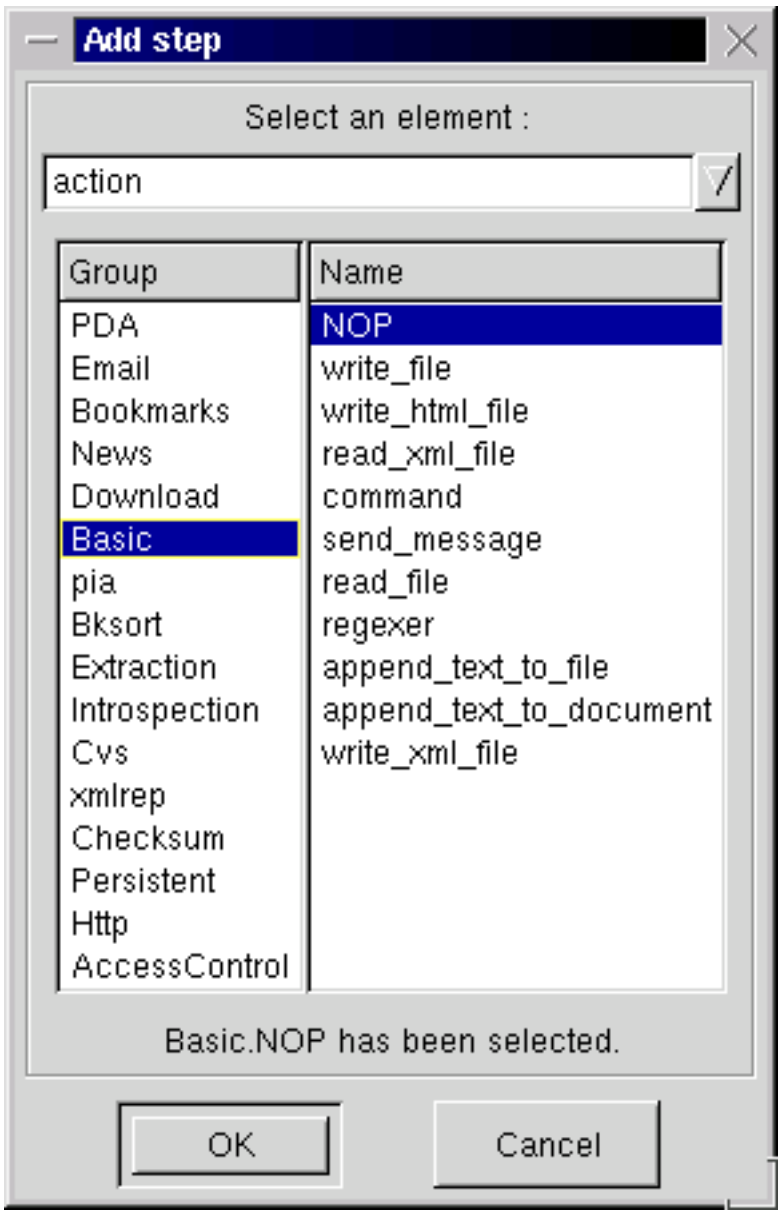

Figure 3. Selection of a step target

b. In the combo box at the top of the window, select "action".
c. In the group list on the left, select the Basic module
d. In the action list on the right, select the NOP action
e. Click on the OK button. This closes the selector, and adds the step to the recipe.

5. Insert the `Basic.read_file` action. This action uses a `URL` element as an input and outputs an `html` element containing the page found at this URL.
   a. select Actions->Add step... in the recipe edition window menu. This brings up a selection window
   b. In the combo box at the top of the window, select "action".
   c. In the group list on the left, select the Basic module
   d. In the action list on the right, select the read_file action

e. Click on the OK button. This closes the selector, and adds the step to the recipe.

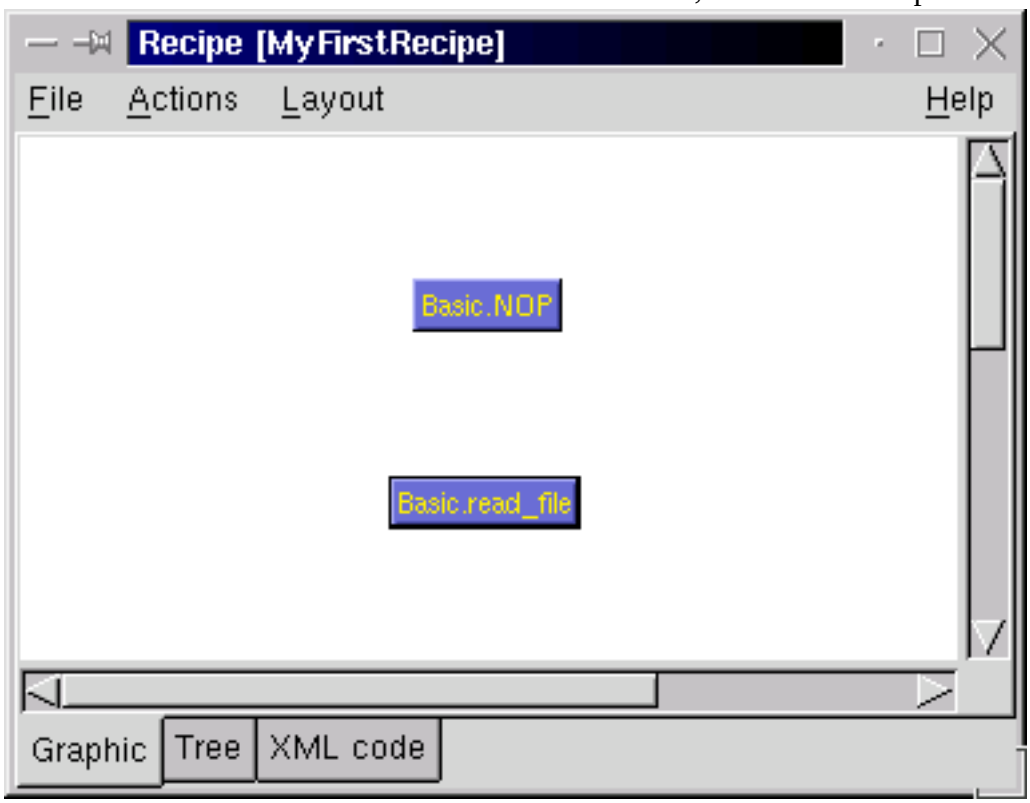

Figure 4. The recipe edition window with the two first steps of the recipe

6. Connect the two steps
   a. Left-click on the `Basic.NOP` step, while holding the Shift key pressed. Notice that the cursor shape changes
   b. Left-click on the `Basic.read_file` step, while holding the Shift key pressed. This connects the two steps and adds the required transition.

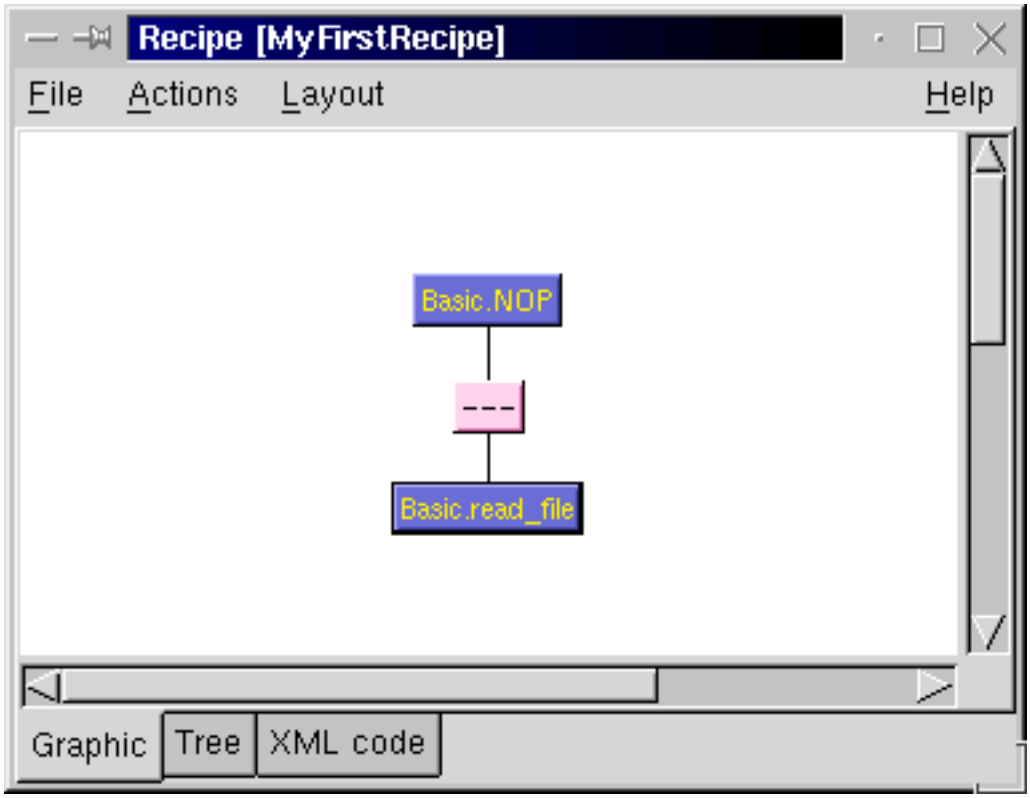

Figure 5. The steps are connected

7. edit the transition properties
   a. Right click on the transition to bring up the pop-up menu
   b. select "Edit transition properties". This changes the edition window view to the tree view, with the transition selected.
   c. Right click on the transition node in the XML Editor to bring up the pop-up menu. This menu allows you to add new children or new attributes to this XML node. You have to add a condition to the transition in order to make it fireable only if a `URL` element is available (else the `Download_html` action will not be able to execute properly).
   d. choose the Append Child->condition menu item. A new `condition` node is inserted in the XML tree.
   e. select this new node and insert a new `match` child with the contextual menu
   f. Select this last node and insert a new `#text` child (in XML Editor, #text stands for classical text).
   g. Select this new node and, in the bottom edition frame, type **`url/text()`**. This means you want your transition to be fireable only if you can get an element that matches the `url` type and contains some text (and this is exactly what you want before executing the `Download_html` action).
   h. With the contextual menu, add a `use` attribute to the condition condition. The new attribute is created with a value of "No".
   i. select the attribute and set its value to **`yes`**. This ensures that the same `url` element will not be used twice to make the transition fireable. This in turn will prevent the associated plans to be executed twice for saving the same html page to

disk.

j. Return to the graphical view of the recipe

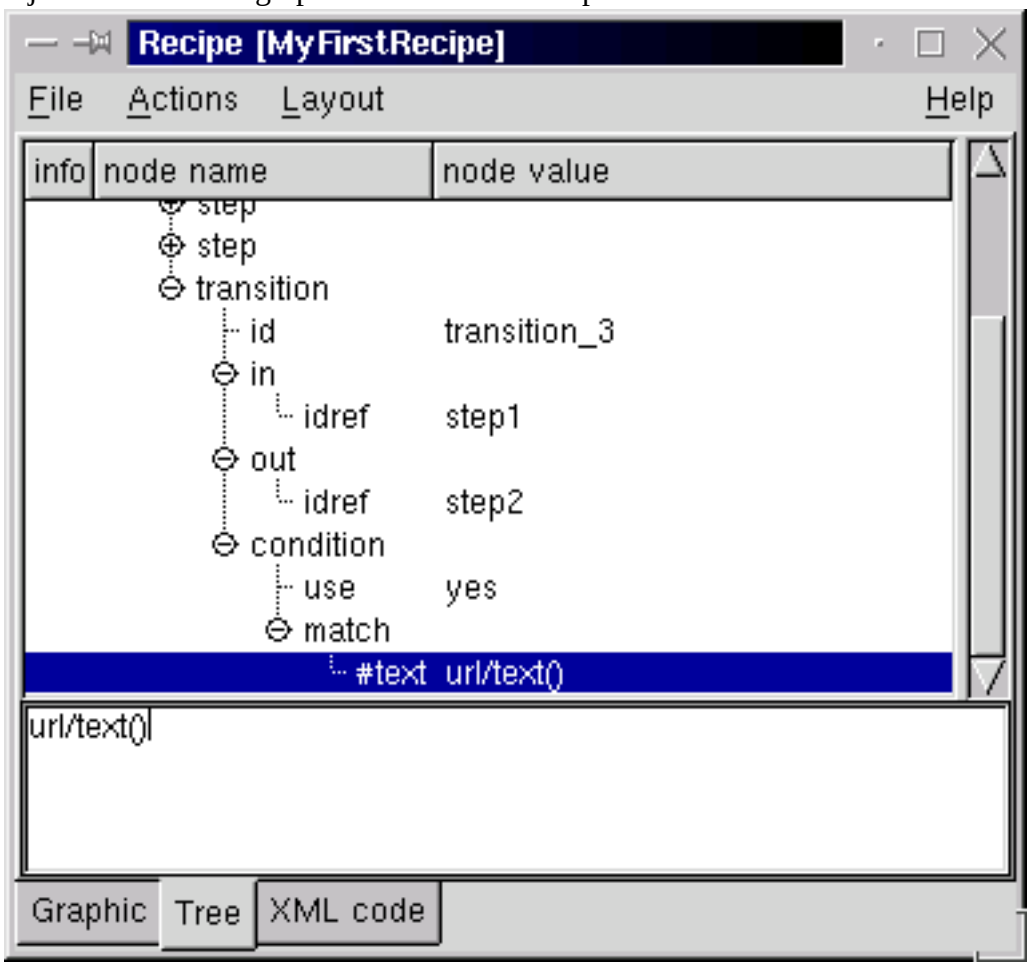

Figure 6. Edition of the transition properties

8. Insert the `basic.change_file_url` transformation
   a. select Actions->Add step... in the recipe edition window menu. This brings up a selection window
   b. In the combo box at the top of the window, select "transformation".
   c. In the group list on the left, select the basic module
   d. In the action list on the right, select the change_file_url transformation
   e. Click on the OK button. This closes the selector, and adds the step to the recipe.
9. Add an explicit argument to the step
   a. Right-click on the newly created step and select edit properties. Horn switches to the Tree view of the recipe with the step selected.
   b. Right click on the `step` node and append `arguments` child with the contextual menu.
   c. Select this new node, Horn automatically opens, in the bottom edition frame, an XML document fragment that contains an empty `argument` node. You can replace this node by XML code describing the elements contained in the action arguments. Be extremely careful, since no validation (except XML syntax) can be

done on the XML code you will enter here, as Horn doesn't know anything about the arguments you want to pass to the action. As argument, you will enter the file name in which the html page will be saved on your disk, for instance `$NARVAL_HOME/data/first_try.html`.

d. replace the **`<arguments/>`** by **`<arguments> <new_url>file:///$NARVAL_HOME/data/first_try.html</new_url> <arguments>`**

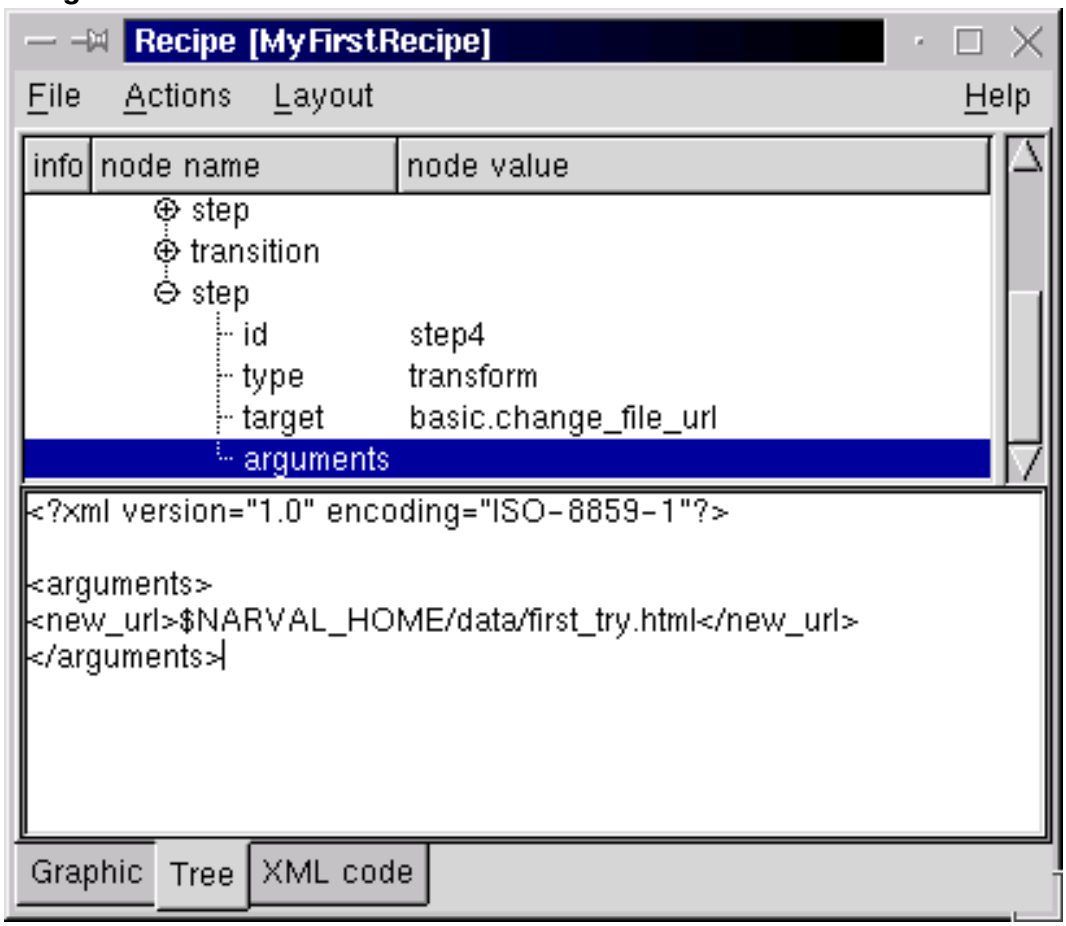

Figure 7. Adding arguments to a step, first stage

e. type Ctrl-Enter to make Horn evaluate the XML code, you have just written. You can see that it has inserted three nested child nodes in the `arguments` node, the inner one (a text node) containing the file name.

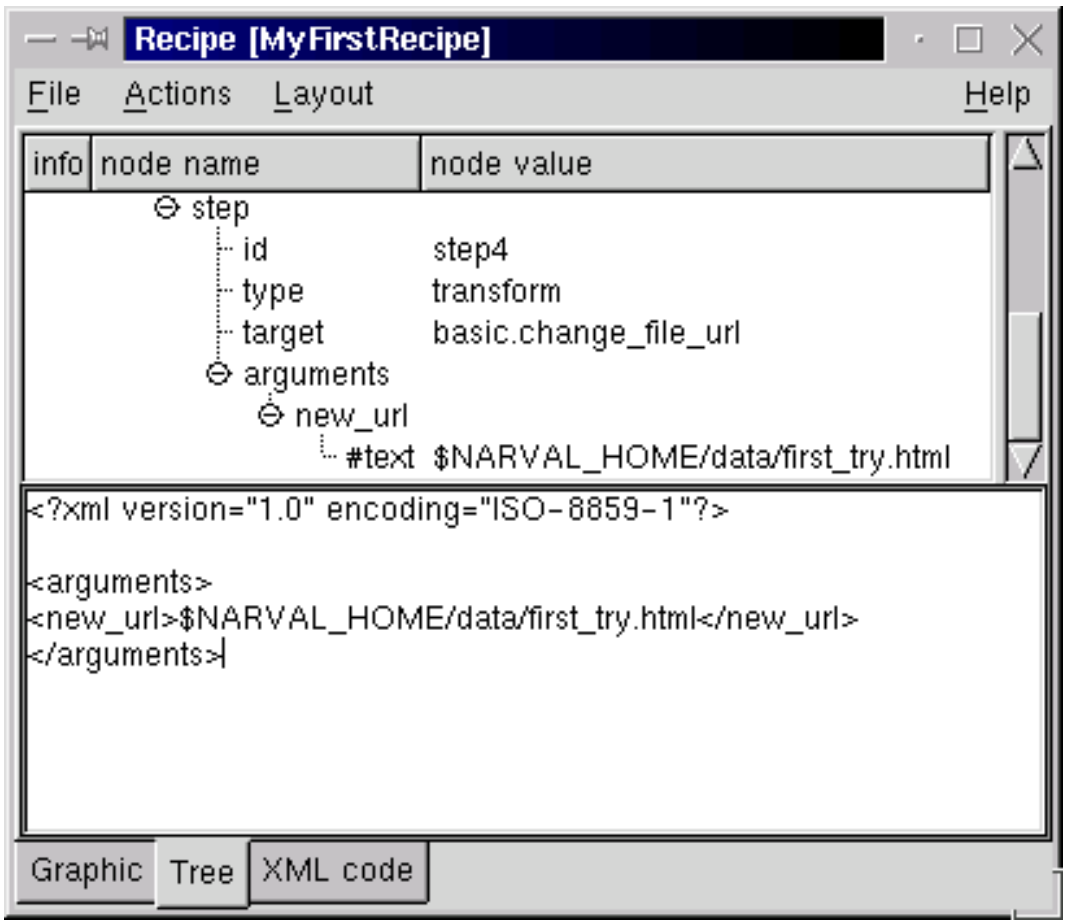

Figure 8. Adding arguments to a step, second stage

10. Connect `Basic.read_file` to `basic.change_file_url`
11. Insert the `Basic.write_file` action
12. Connect `basic.change_file_url` to `Basic.write_file`

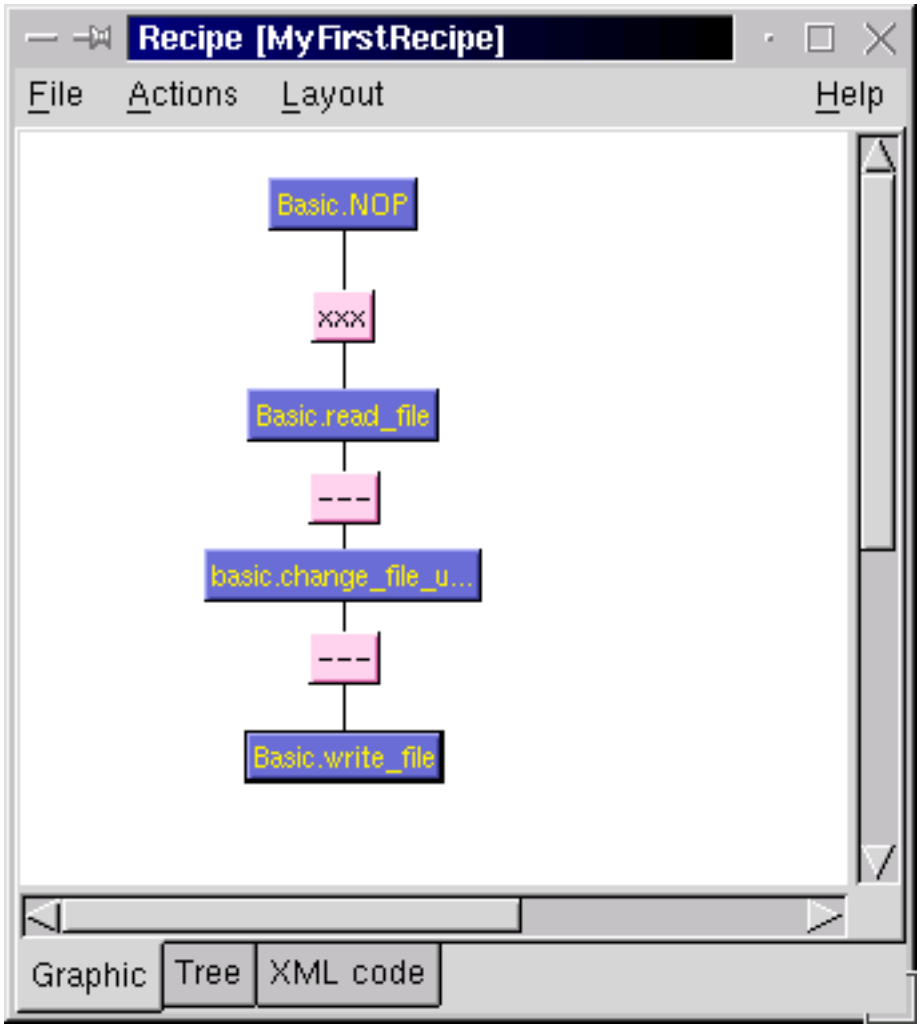

Figure 9. Almost done

13. Edit the properties of the recipe
    a. select the Actions->Edit properties menu item and choose the start and end steps. Start step is `Basic.NOP` and end step is, of course, `Basic.write_file`. You must select the steps through their id. If you don't remember them, you can find them in the tree view of the recipe.
    b. check the autorestart checkbox so that you plan will automatically restart

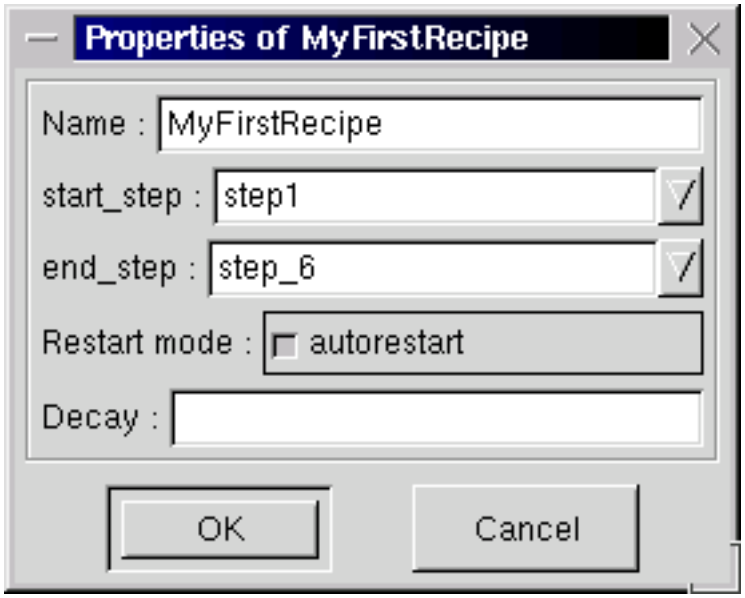

Figure 10. Edition of the recipe properties

14. save your first recipe with the File->Save menu item.

6. Running your recipe

First of all, it is necessary to make sure that Narval knows a few things about your network configuration. The file `$NARVAL_HOME/data/memory.xml` holds some configuration information such as your email address of the name of your proxy. These are set to default values upon installation, and you should edit them if they do not match your settings (which is quite possible, especially for the email address). If you have a direct connection to the Internet, you should remove the line describing the proxy, which is <proxy type="http" hostname="proxy" port="3128"/> by default.

Launch Narval. Then, in Horn, use Narval->Connect... to connect to Narval. Check Options->Monitor plans menu option so that new plans will be automatically monitored.

In the memory window, right click on my_recipes.MyFirstRecipe and select Instanciate plan in the pop-up menu.

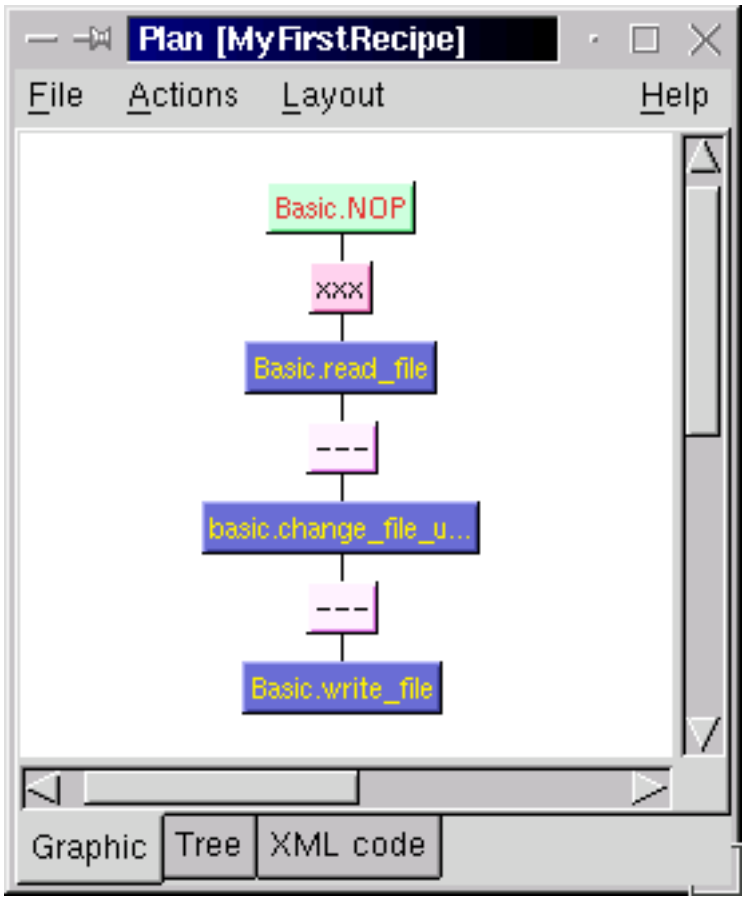

Figure 11. The Plan is waiting for an element

The plan supervision window is displayed and you can see that the execution cannot pass the first transition. Why? If you recall, you specified that this transition could only be fired if an element matching url/text(), and no such element is in memory. Well, this is easy to fix: in the memory window, use Interact->Add element to bring up a small text editor in which you will be able to enter the missing element. For instance, you could use **`<url>http://www.logilab.org/narval/index.html</url>`**.

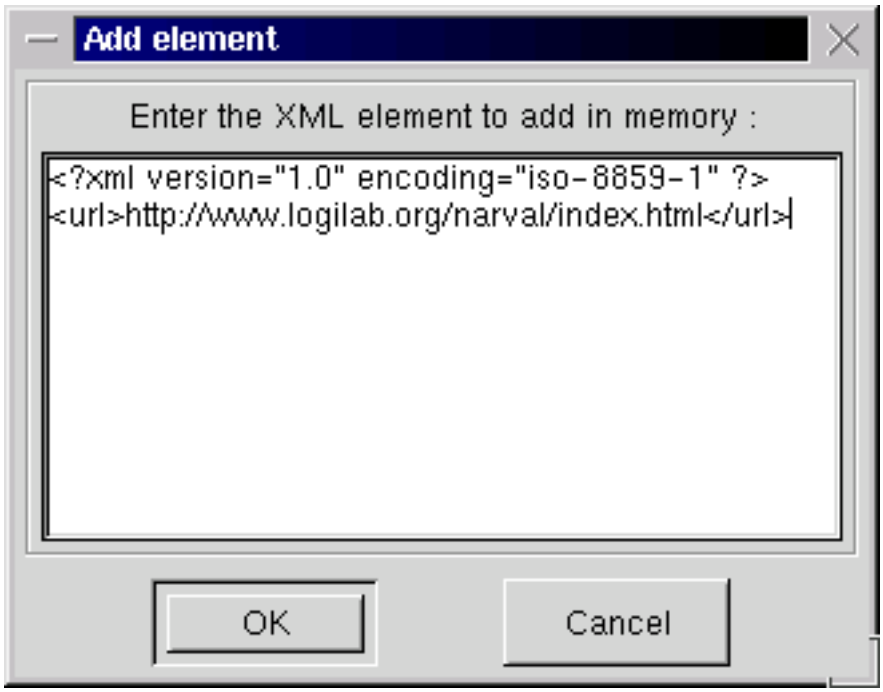

Figure 12. Adding an element in the memory

When you hit the OK button, the element is sent to Narval, and the transition is fired. When the plan has been successfully executed, you can check that the file you downloaded was successfully saved to disk.

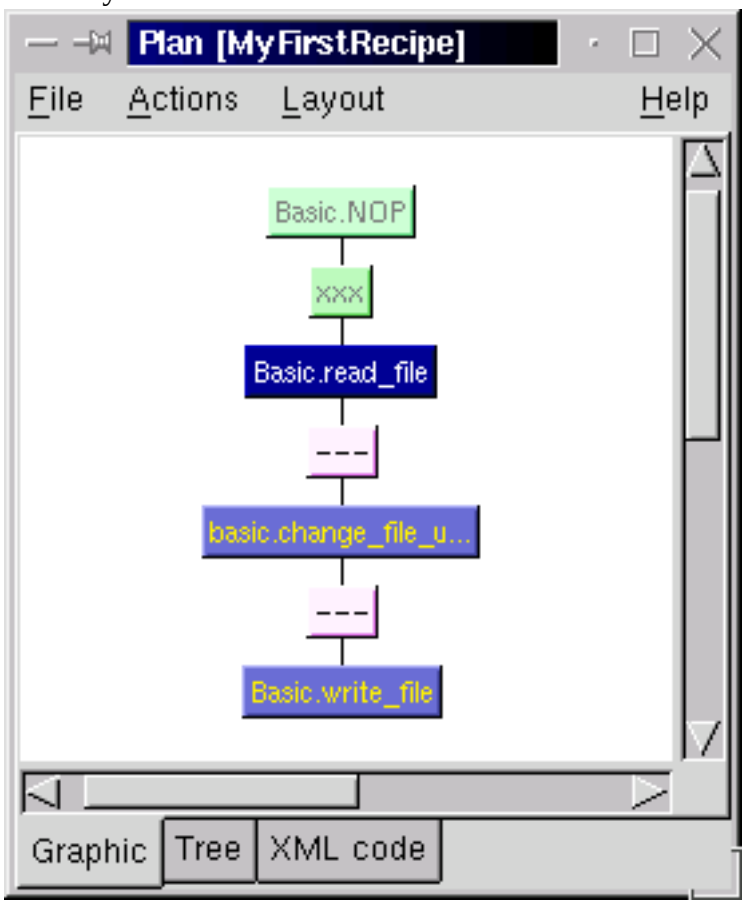

Figure 13. Plan execution can go on

There you are! Your first recipe is running. We hope that you have enjoyed this step by step walk through the construction of a recipe. The goal here was to show you the basic functionnalities of Narval (steps, transitions, actions, conditions, arguments) and a couple of actions and transformation from the standard library. There are a lot more features in Narval than that, and you certainly have lots of questions on how such and such functionnality can be achieved with Narval. The mailing lists are the right place to ask such questions, report bugs in the documentation (or the code of Narval), discuss features you'd like to see in the language. Feel free to use them. We'll do our best to help you, and so will other people using Narval.