Narval User Manual

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

Alexandre Fayolle
Olivier Cayrol

Narval User Manual

by Alexandre Fayolle and Olivier Cayrol

Legal Notice

Abstract

Narval (Network Assistant Reasonning with a Validating Agent Language) is a language and an interpreter for that language. The language is well adapted for setting up intelligent personal assistants. This document explains the objectives and the general use of Narval. It describes Horn (Human Oriented Ressources for Narval), its user interface (graphical and command line remote control), and contains also informations about Narval installation.

Logilab

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

There is a fundamental need of a unique and intelligent interface able to unify the various accesses to information, and to ease the interactions between different kinds of elements, and overall to understand the user intentions expressed in a high-level language closer to natural language, while hiding all the technical low-level operations.

1.1. Narval, a new paradigm

Logilab has designed Narval. Narval stands for Network Assistant Reasonning with a Validating Agent Language. More than a new classical software, Narval is, first of all, a new paradigm that might change the relationship between people and technology. Taking advantage of the latest techniques used in artificial intelligence and the most recent works in computer science, Narval can be seen as the user personal assistant: it lives in the information world and is able to use all the resources, all the documents and all the programs it finds in order to satisfy the needs expressed by the user. Moreover, it is able to understand what it is doing, which allows it to detect its own errors but also to have an intelligent exchange with the user for describing its actions and their results.

Narval is an original computing engine that can execute user defined tasks. To do this, Narval behaves as a conductor piloting the existing softwares and computer devices in order to perform each of the necessary elementary actions. Therefore, for instance, to download the weather forecast every morning (as the user asked to), Narval will run the Internet connection, get the web site address from the bookmarks, download the correct web page located at this address, save it on the disk and make it available from the browser home page.

Narval's philosophy is not to replace the existing software tools but to integrate all their functionalities to ease their execution and their sequencing while hiding complexity.

Chapter 2. Design philosophy

2.1. Generic approach

2.1.1. Narval, a ready-for-use product

Although it uses some of the latest design techniques in computer science, Narval is, first of all, a robust and functional software. It is not a research prototype used for evaluating various software implementations of an algorithm. Therefore, even if it is possible to enhance its performances (CPU and memory usage), most of Narval future development will be driven by new applications. Narval purpose is to be the user's unique assistant for the access to informations world, so it has to be interfaced with WAP cellular phones, personal digital assistants, etc. Narval releases will mainly contain new functionalities useful to the end consumer.

2.1.2. Narval, an easy-to-use software

Narval has been designed keeping in mind it will be used by people non familiar with computers. Its main objective is to simplify access to information. Thus, the graphical interface, the tasks Narval can run or the name of the elements it can handle have been designed and chosen to be easily understood even by a neophyte. The same spirit will animate Narval future developments. Hence, even if it could be possible, Narval will not become a graphical high-level programming language. Narval must remain a simple tool usable by all the kinds of users, even the ones with very little experience in computing.

2.1.3. A few more technical points

Narval relies on a rule-based system. This provides reactivity as well as modularity. Furthermore, this rule-based system has been crossed with a plan-based system in order to provide elementary steps sequencing. The result conceptually fits to most situations Narval will have to manage. Thus, it will not evolve towards a script running software managing state machines or Petri nets.

2.2. Comparison with other computing science concepts

2.2.1. Rule-based systems

Rule-based systems are a class of systems where elementary rules are made available to the program. These rules give instructions on how to obtain a result given a set of preconditions, and they are generally not organized. The program is then given a final goal, and it attempts to find which rules to apply, given this goal and what it already knows to achieve the goal. A simple and well known example of such system is the make program, which is able to run different compilers to produce an executable piece of machine code for instance.

Such systems are highly reactive, since the program constantly evaluates what it should do next and has no predetermined route, and are perfectly designed for goal-oriented programming.

Narval borrows from these systems, in that recipes can be written to wait for an event to happen and process it. To keep on with the makefile example, we could have a recipe that would be able to turn a C file into a object file, and another recipe that would wait for a set of object file to be present in memory and link them to produce an executable.

2.2.2. Plans

Another class of systems uses plans. In these systems, the elementary actions are preassembled into plans which describe in detail how to achieve a given goal. These systems are generally less flexible than rule-based systems, but tend to be more effective when performing dedicated tasks.

Narval has such features, since it is based on recipes, which basically decribe how a task should be performed. A recipe can be as simple or as complex as required, thus letting the user control how rigid a behaviour he wants.

2.2.3. Contract based programming

A large trend of computer science deals with contract programming, either at the specification level (such as UML's Object Constraint Language), or at the implementation level (such as Eiffel pre and post conditions mechanism). To sum things up, these are means to ensure properties of arguments and return values, beyond the mere type of the data. When writing a standalone program, you are in full control of what arguments are passed to functions, and what these functions return, but when designing a function library, this is not possible. Many libraries use assertions to ensure that the arguments do have these additionnal properties, but the end user has to refer to some external documentation to check these. In languages such as Eiffel, the compiler adds the necessary checking code.

The Narval interpreter spends lots of efforts ensuring that the inputs of actions have the required properties, and that the outputs are conform to what the action said they would be.

Chapter 3. Narval generic behaviour

3.1. The memory

The most important part of Narval is its memory. It contains all the elements Narval handles or might handle. Some examples of elements are: an email, a document, a web page, etc. An element can also be a plan or a recipe (both notions will be defined in further sections).

> **Remember**
>
> Anything Narval can handle is an element. Each element is stored in Narval's memory.

3.2. Recipes

The only thing Narval can do with the elements found in the memory, is apply recipes. A recipe is a sequence of actions performed on elements. There are, for instance, recipes for automatically sorting and answering emails, or checking out the new Web pages inserted on a given Web site.

Please note that a recipe are not executed by Narval: a recipe is a specification kept in memory. Executing a recipe requires the creation of a plan (see next section).

> **Remember**
>
> Narval uses recipes to manipulate elements in memory. A recipe specifies a sequence of operation performed on, but it can not be executed.

3.3. Plans

A plan is a very special element: it corresponds to a recipe being executed and can manipulate or create elements in the memory. In computing terminology, a plan is an instance of a recipe. After its execution is finished, the plan is destroyed.

In the same way as recipes, plans are sequences of elements manipulation instructions. To be more precise, the sequences are made of steps and transitions(see next sections).

> **Remember**

> A plan corresponds to a recipe being executed.

3.4. Steps

Within plans and recipes, steps are the elementary way of specifying a manipulation of elements in memory. Passing from a step to another is done through transitions (see next section).

> **Remember**
>
> A step specifies an elementary manipulation of elements in memory.

3.5. Transitions

Within a plan or a recipe, how to pass from a step to another is described by transitions. A transition can have several incoming steps and several outgoing steps: when all the incoming steps have been executed, the transition can be triggered, the outgoing steps are then executed in parallel.

A transition can have a condition. In that case, it can not be triggered unless the condition is true.

> **Remember**
>
> A transition allows passing from a step to another (or from several steps to several others). A transition can have a condition and, isn't fired unless the condition is satisfied.

3.6. Actions

An action is always associated with a step. It is one way of manipulating an element.

An action is linked on one side to a prototype describing the elements it takes in and the elements it outputs, and on the other side to the code of the manipulation, written in Python.

The same action can be used in different steps and different recipes.

> **Remember**
>
> An action is associated with a step and is a way of manipulating elements in memory.

3.7. Modules

A module contains several actions that are conceptually close by their functionalities or their code.

Remember

A module is a group of actions.

3.8. Transformations

A transformation is always associated with a step. It is another way of manipulating data in Narval's memory.

A transformation does not involve Python code, but only structural manipulation of the element. The main consequence is that it cannot have any effects outside the Narval interpreter, such as changing the contents of a file.

Remember

A transformation is associated with step and is a way of handling of the elements in memory.

Chapter 4. Running Narval

4.1. Command line syntax

Narval has a number of command line switches:
`narval` [--debug] [--home *`home-directory`*] [--quit-when-done] [--quit-after *`seconds`*] [--start-plan *`cookbook.recipe-name`*...] [--listen-on *`port`*] [--socket-manager] [--max-threads *`nbthreads`*] [--load-memory-file *`file`*] [--save-mem-on-quit] [--quit]

The options that are often used are:

--home home-directory

Narval uses the provided directory as a home directory. If the switch is not passed, the `NARVAL_HOME` environment variable is used. If it is not set, then, on Unix platforms `$HOME/.narval` is used as a default. Be sure to pass a directory where Narval will find all the data it needs, especially when running in unattended mode, since Narval will ask you if it should set up the directory the contents are not what is expected. A configuration file `/etc/narval.conf` is installed with Narval and it contains information on the location of some data files for Narval.

On Windows systems, the registry is used to hold configuration data. The default `NARVAL_HOME` is `\\HKCU\Software\Windows\CurrentVersion\Ex Folders\AppData\narval_data`. There is no `narval.conf` file. Equivalent information can be found in `\\HKLM\Software\Logilab\Narval\VERSION\co` Each user can override the global settings and use his own by populating `\\HKCU\Software\Logilab\Narval\VERSION\co`

--quit-when-done

Narval will quit when all plans have reached *done* or *failed* states. This flag is used with the --start-plan flag

--quit-after N

Narval will quit after N seconds of execution. This flag is used with the --start-plan flag, especially if the started plan is in restart mode.

--start-plan PLAN	Narval will execute PLAN right after starting
--listen-on PORT	listens for UI client connections on specified port. This must be set if you intent to connect to Narval from Horn.
--socket-manager	starts the socket manager service. This must be set if you intent to run recipes that use the socket manager, such as the proxy or the PDA.

The following options are less commonly used. Most of them are used for debugging, testing or tuning, but they may be useful in other contexts.

--debug	prints additional messages during execution. Only use this if you have a problem, since it will slow down the execution.
--max-threads N	sets maximum number of concurrent threads to N. The default is 10.
--load-memory-file	uses specified FILE instead of the default memory file, which is data/memory.xml in Narval's home directory.
--save-mem-on-quit	Narval will save memory to FILE when quitting. This can be used for diagnostics.
--help	prints a usage message.

Chapter 5. Horn, Narval user interface

5.1. General presentation

HORN (Human Oriented Ressources for Narval) is the user interface for Narval. Two interfaces are available : a graphical user interface - GUI and a command line interface (CLI). There are also two modes available when using Horn: you can be connected to a running Narval, which will enable you to control the interpreter by launching plans, edit Narval's memory, supervise the execution of plans, or you can run Horn locally, for instance to create new recipes and cookbooks.

5.2. Launching Horn

5.2.1. Launching the graphical interface (GUI)

To launch GUI mode, enter the following command: **horn** or **horn --gui**

5.2.2. Launching the command line interface (CLI)

To launch CLI mode, enter the following command: **horn --cli**

5.2.3. More command line options

There are others command line options for Horn:

--home home_path	Specify a new `NARVAL_HOME`
--exec cmd 1,...,cmd n	Execute a set of commands in Horn CLI mode.
--execfile file	Execute a script of CLI commands line.

5.3. Using the graphical interface

5.3.1. Connection window

When you start Horn with the graphical interface, the connection window is displayed. The

following options are available: connect to a running Narval (which must have been launched with the --listen-on command line option), work with data from a local `NARVAL_HOME`, or do nothing special.

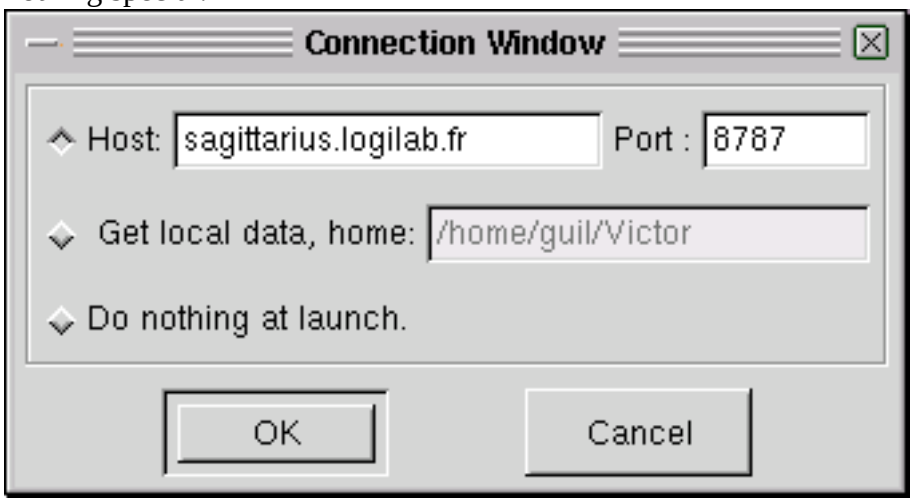

Figure 5.0. Connection window

If you choose to run Horn in connected mode, the default host name is the name of your computer, and default port is 8787. You should set these to values matching the location of the Narval server and the port on which it is accepting connections. If you choose in local mode, default HOME is `$HOME/.narval/` on unix systems, or `app_dir\narval_data` on Windows, where app_dir is the value of the application data directory as read from the registry, unless you specified an other HOME on the command line.

If you specify an explicit `NARVAL_HOME` that does not contain the required information (important files are missing, for example), Horn displays a dialog window prompting you to copy missing files from `$NARVAL_SHARE` or from localisation path written in configuration file `/etc/narval.conf` on Unix systems or from the registry on Windows if environment variable does not exist...

5.3.2. Horn main window

The main window is made of a menubar, a notebook and a status bar.

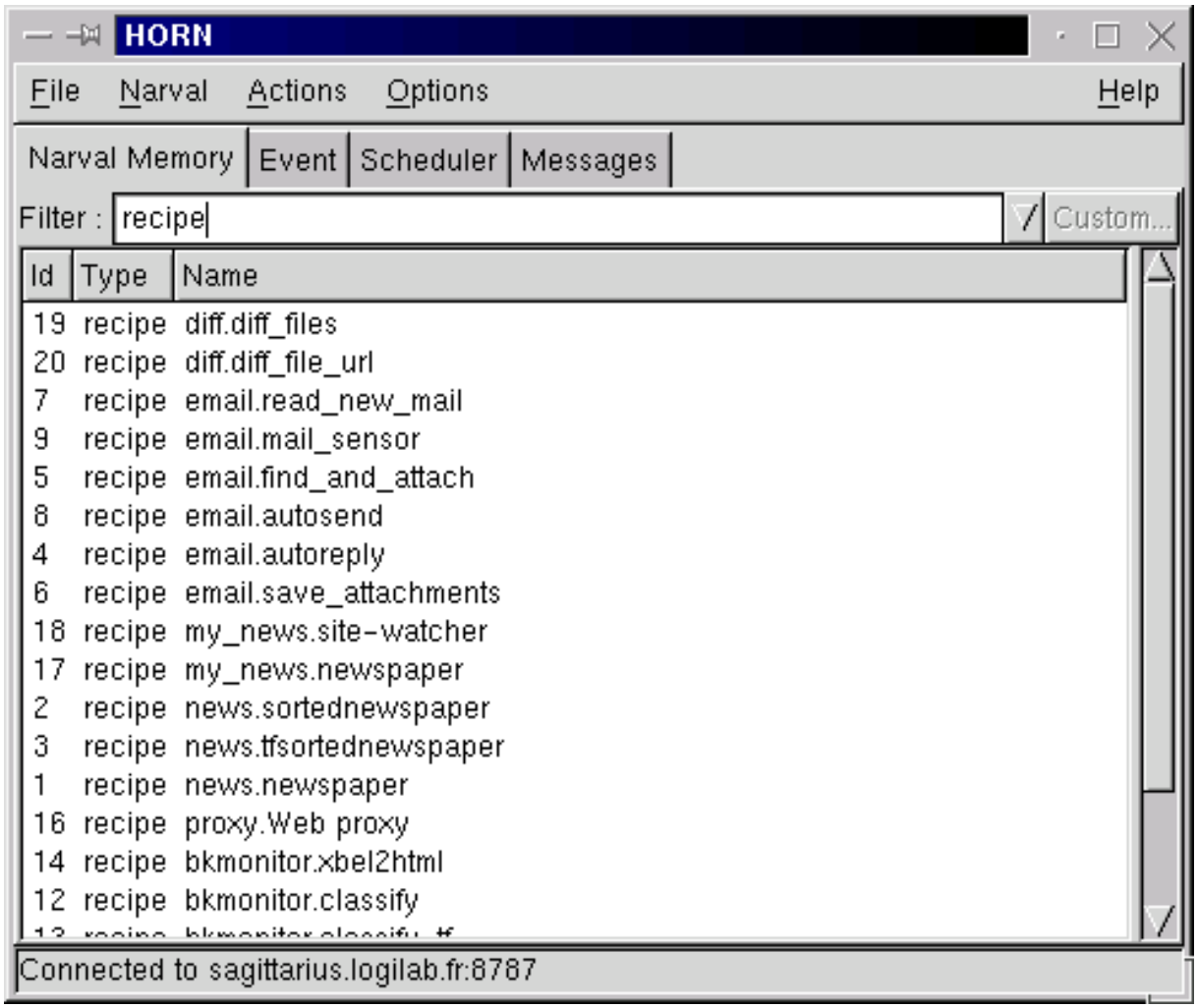

Figure 5.1. Main window, in connected mode

5.3.2.1. The File menu

The File menu is used control the settings in local mode. Here is a description of the menu items:

File->Narval home... — Available in local mode. Displays a directory selector window, which enables the selection of the `NARVAL_HOME` directory. If the directory does not have the required subdirectories, a dialog is brought up so that the user can decide to create empty subdirectories, to create subdirectories with default contents obtained from the NARVAL_SHARE directory or reconsider his choice.

File->Install package... — Available in local mode. Displays a dialog window to select a npm package (Narval Package Manager). this package will be install in current NARVAL HOME.

File->Uninstall package... — Available in local mode. Displays a dialog window to select a manifest file. This file contains

	data to uninstall corresponding package from current NARVAl HOME.
File->Quit (**Ctrl-Q**)	Exits the graphical interface.

5.3.2.2. The Narval menu

The Narval menu is used to pilot an instance of Narval through the graphical interface. Here is a description of the menu items:

Narval->Connect...	Available in local mode. Establishes a connection with a remote Narval instance.
Narval->Disconnect	Available in connected mode - Terminates the connection of the graphical interface with the remote Narval instance.
Narval->Suspend	Available in connected mode - Suspends the execution of the Narval instance to which the graphical interface is connected.
Narval->Step	Available in connected mode - Asks the Narval instance to which the graphical interface is connected to perform an elementary operation, such as evaluate a transition or launch an action.
Narval->Continue	Available in connected mode - Continues the execution of the Narval instance to which the graphical interface is connected normally.
Narval->Save recipes	Available in connected mode - Saves the recipes in Narval's memory to the files where they were read from. Narval should have write access on these files.This is a good thing when you are editing the recipe in real time when debugging. If problems occure while saving, Narval will create an error element in memory.
Narval->Save transforms	Available in connected mode - Saves the transforms in Narval's memory to the files where they were read from. Narval should have write access on these files. This is a good thing when you are editing the transformation in real time when debugging a recipe. If problems occure while saving, Narval will create an error element in memory.
Narval->Start plan...	Available in connected mode - Displays a recipe selection dialog in which the user can select a recipe to instanciate.
Narval->Add element...	available in connected mode. Displays a small text editor in which the user can type the XML representation of an element which will be added to the memory of Narval.

5.3.2.3. The Actions menu

The Actions menu is used to manipulate elements like recipes, cookbooks, action and transformations. There are also clean commands for event, scheduler and messages pages:

Actions->Save cookbook...	Available in local mode. Displays a dialog window to select cookbook and save it.
Actions->Save all cookbooks	Available in local mode. Save all cookbooks into current `NARVAL_HOME`.
Actions->New recipe...	Available in local mode. Displays a dialog window to create a new recipe. For more details, see Section 5.9.1.
Actions->Edit recipe...	Displays a dialog window to choose a recipe and edit it. For more details, see Section 5.9.2.
Actions->Remove recipe...	Available in local mode. Displays a dialog window to choose a recipe and remove it.
Actions->Save recipe...	Available in local mode. Displays a dialog window to choose a recipe and remove it.
Actions->Edit action...	Displays a dialog window to choose an action and edit it.
Actions->Remove action...	Displays a dialog window to choose an action an remove it
Actions->Edit transformation...	Displays a dialog window to choose a transformation and edit it.
Actions->Remove transformation...	Displays a dialog window to choose a transformation and remove it.
Actions->Clean Event Queue	Available in connected mode. Cleans the Event Queue page messages. For more details, see Section 5.5.
Actions->Clean Scheduler Queue	Available in connected mode. Cleans the Scheduler Queue page messages. For more details, see Section 5.6.
Actions->Clean Messages List	Available in connected mode. Cleans Message List page messages. For more details, see Section 5.7.

5.3.2.4. The Options menu

The Options menu is used to provide access to the configuration of several options of the graphical interface. Here is a description of the menu items:

Options->Monitor plans	When this item is enabled, a new plan supervision window is displayed each time a new plan is launched.
Options->Preferences	Opens the application preference edit dialog. The preferences available for edition are described in Section 5.8.

5.4. Memory view

This section is about memory view in Horn. The memory view displays a list of elements that can be used to perform various operations. In local mode, these elements are collected from data files in NARVAL_HOME. In connected mode, these elements are those available in Narval's memory, and the view mirrors changes that occurs in Narval's memory.

Memory view is a list. Right clicking on an element pops up a a menu which offers various operations depending on the element. It is possible to filter the displayed elements using the Filter combo box. It is also possible to sort the list by cliking on the column name buttons.

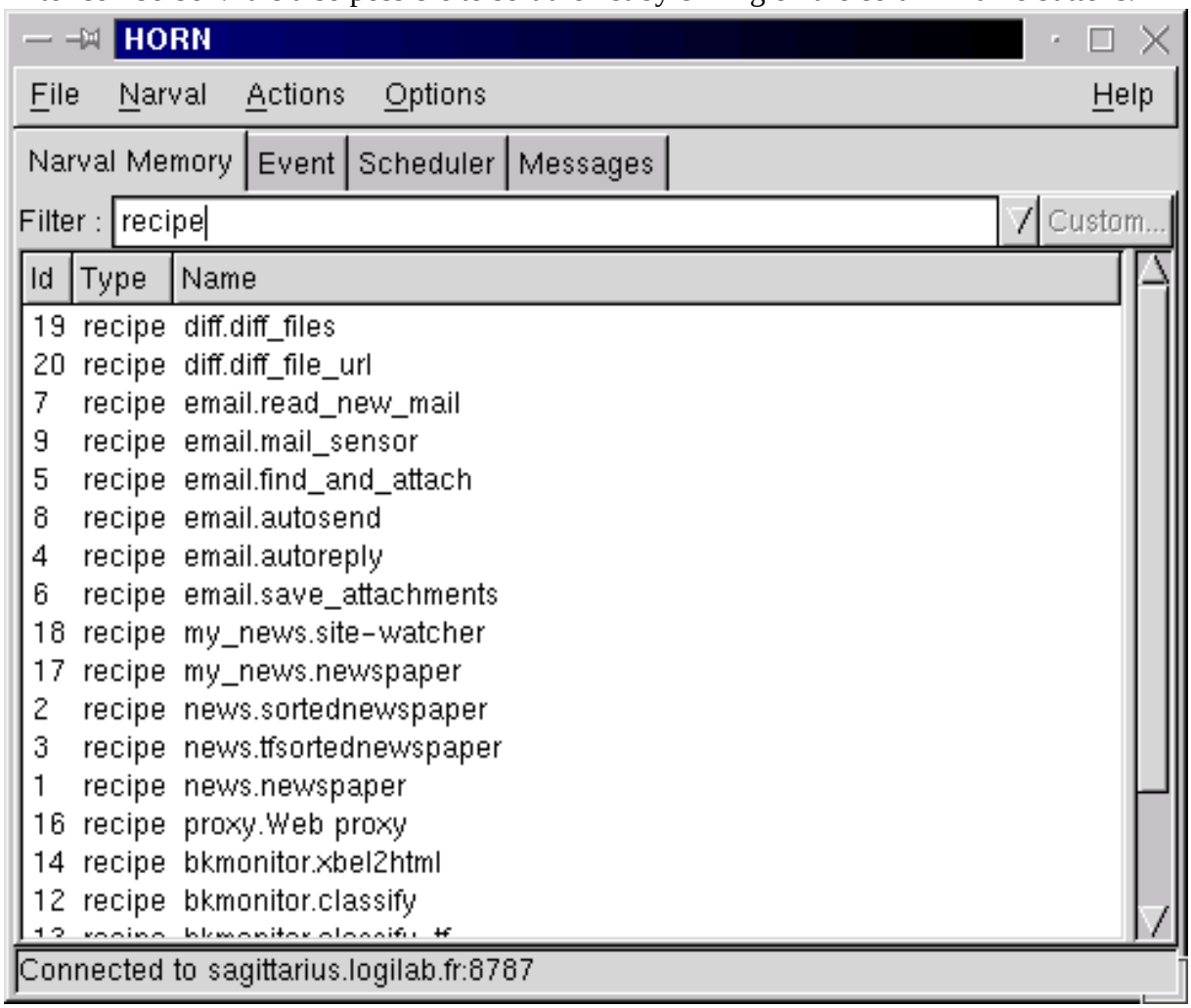

Figure 5.2. Memory view

5.5. Event Queue view

Available in connected mode. This view displays the events from Narval's work activity. It

displays the internal event queue of Narval. Once a element is removed from the queue by Narval, the color changes in the view. It is possible to clear the list using the clean command in the Actions menu.

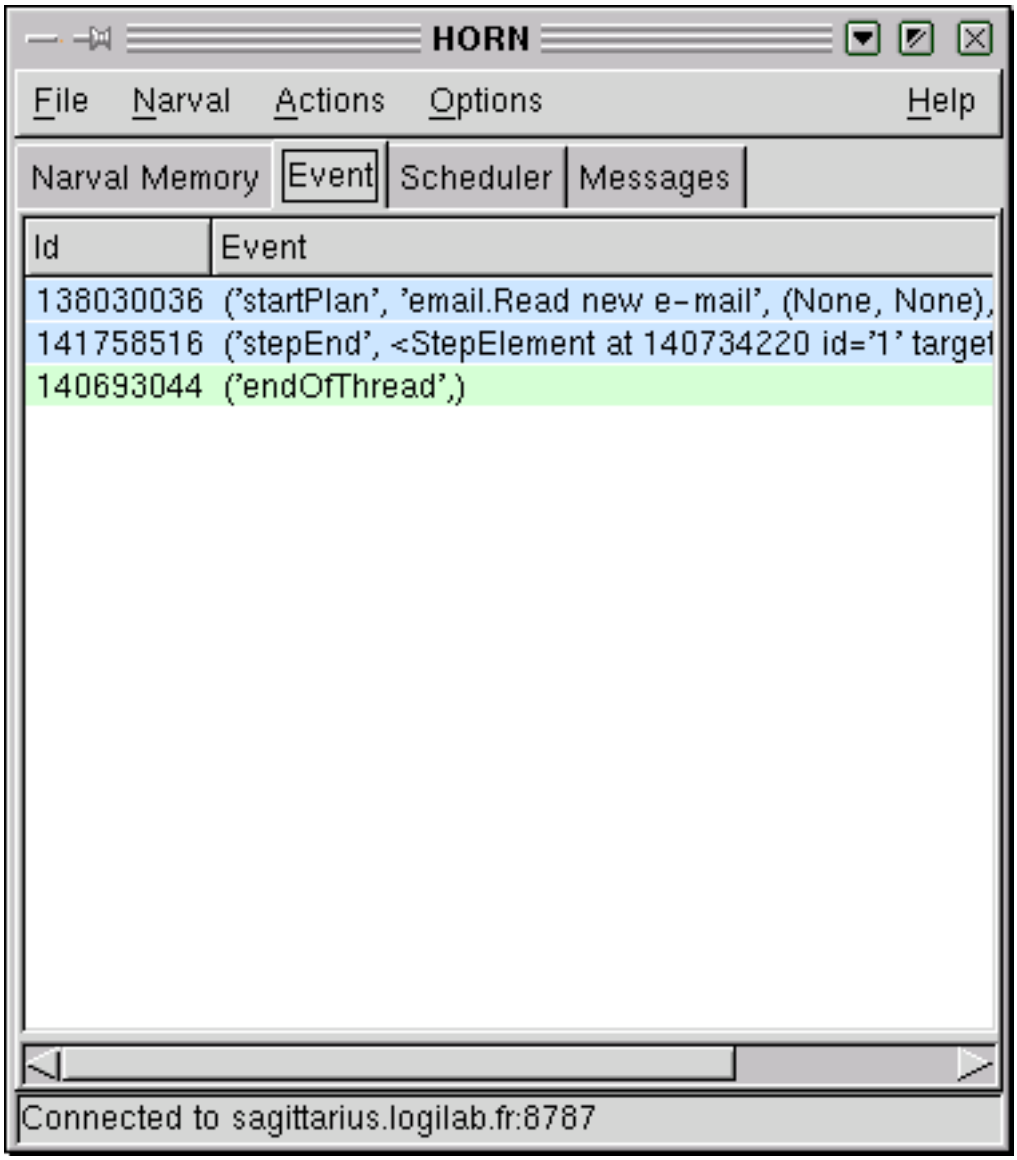

Figure 5.3. Event Queue view

5.6. Scheduler Queue view

Available in connected mode. This view displays pending commands that will be executed later by Narval. Once a pending command has been processed by Narval, the color changes in the view. It is possible to clear the list using the clean command in the Actions menu.

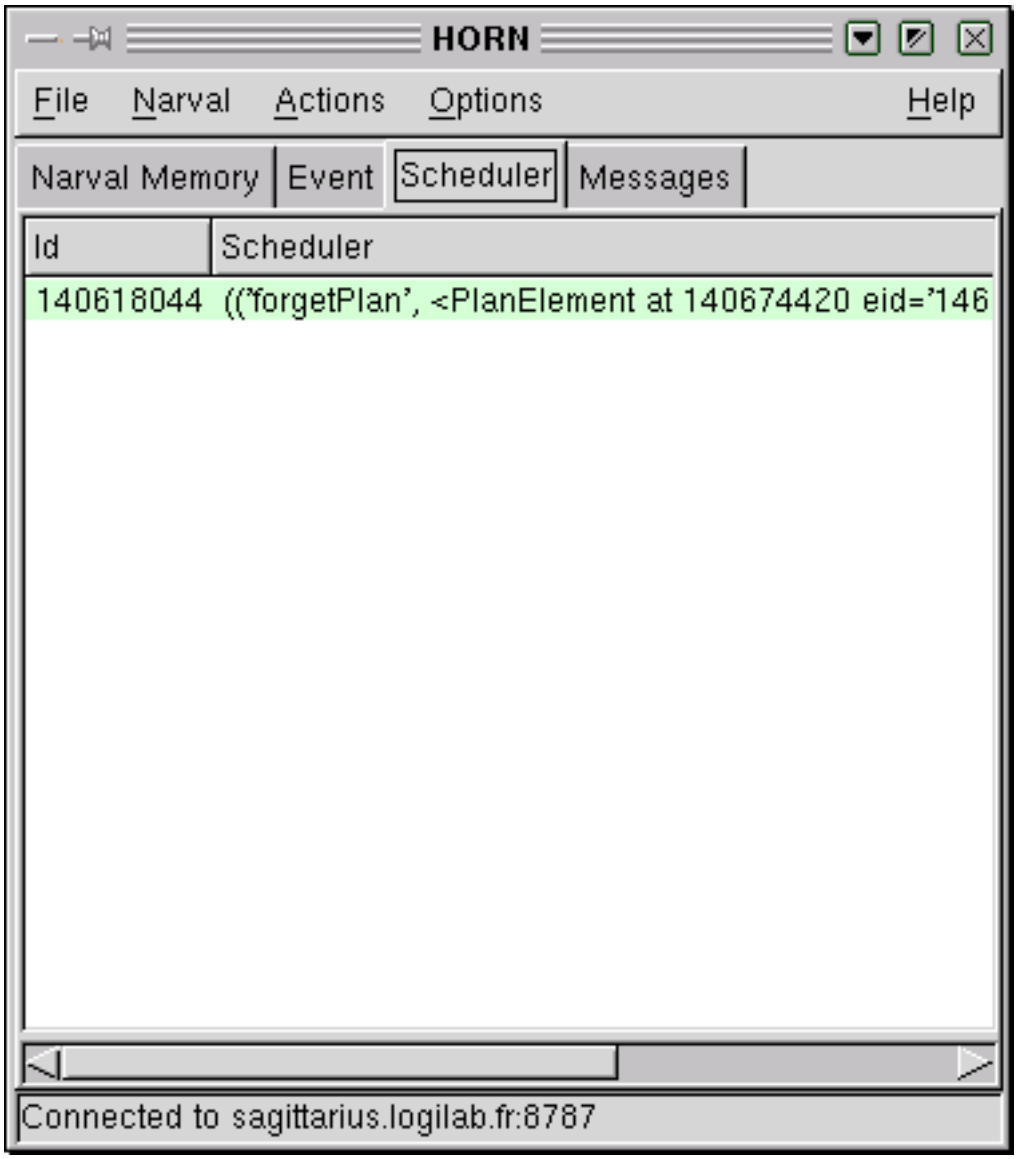

Figure 5.4. Scheduler Queue view

5.7. Messages List view

Available in connected mode. This view displays the list of messages sent by Narval to Horn. It is possible to clear the list using the clean command in the Actions menu.

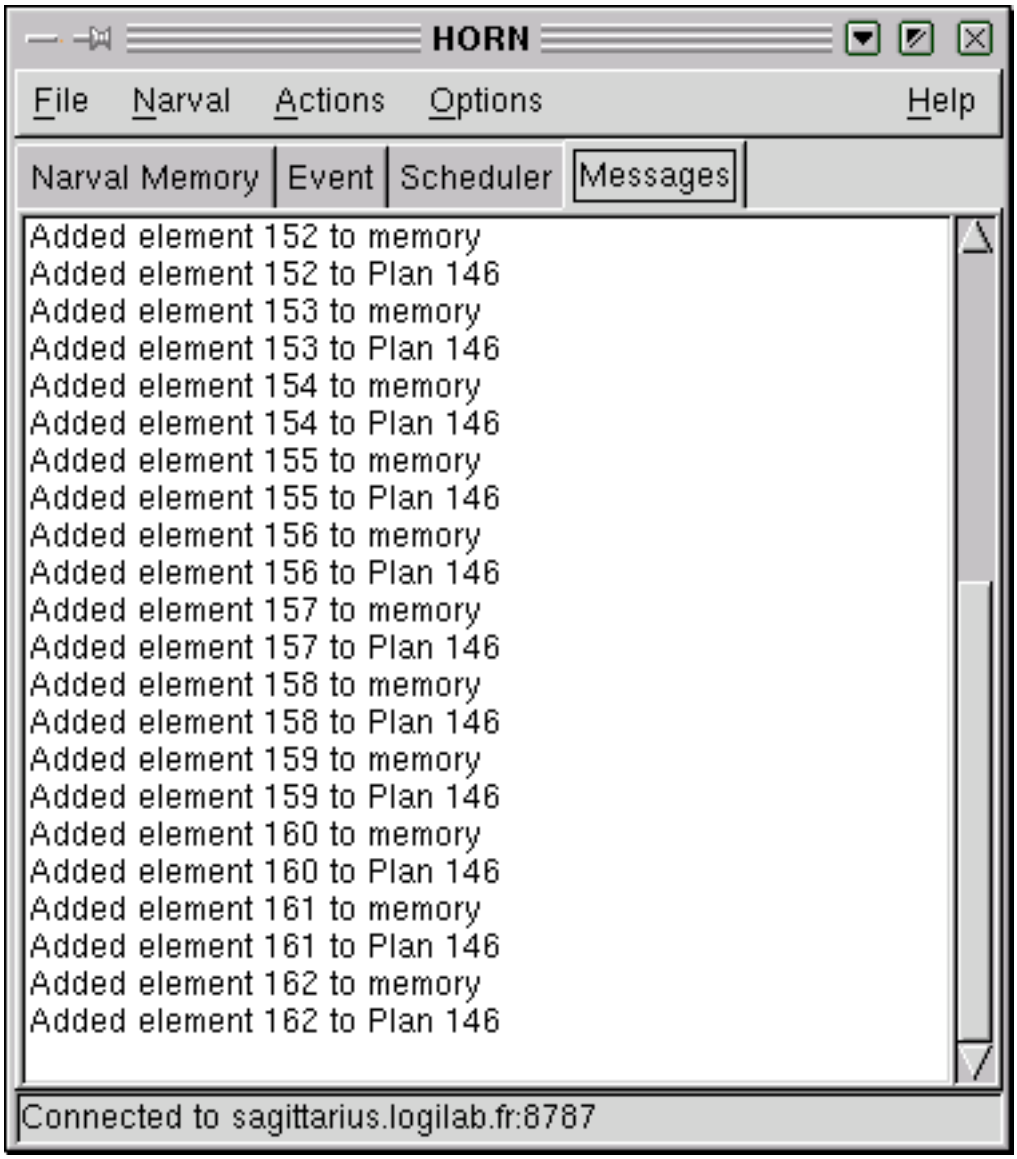

Figure 5.5. Messages List view

5.8. Editing the preferences

5.8.1. Aspect of a plan element during execution

The Recipe Display Style tab of the preference edit window (see Figure 5.6.) enables to change colors indicating the state of steps and transitions during the execution of a plan. A preview of the aspect is given in the bottom right corner of the dialog. here is a description of the editable fields:

Element type — the dropdown list allows to choose the category of elements to edit. Available categories are: step, transition.

State	the dropdown list allows to choose the element state for which the aspect is edited. Available states depend on the selected category. For more information about the differents states, please refer to the Narval Technical Manual.
Font...	opens a font chooser to choose the font of the element label.
Font color...	opens a font chooser to choose the color of the font of the element label.
Border color...	opens a color chooser to choose the color the border of the element frame.
Background...	opens a color chooser to choose the color of the background of the element frame.

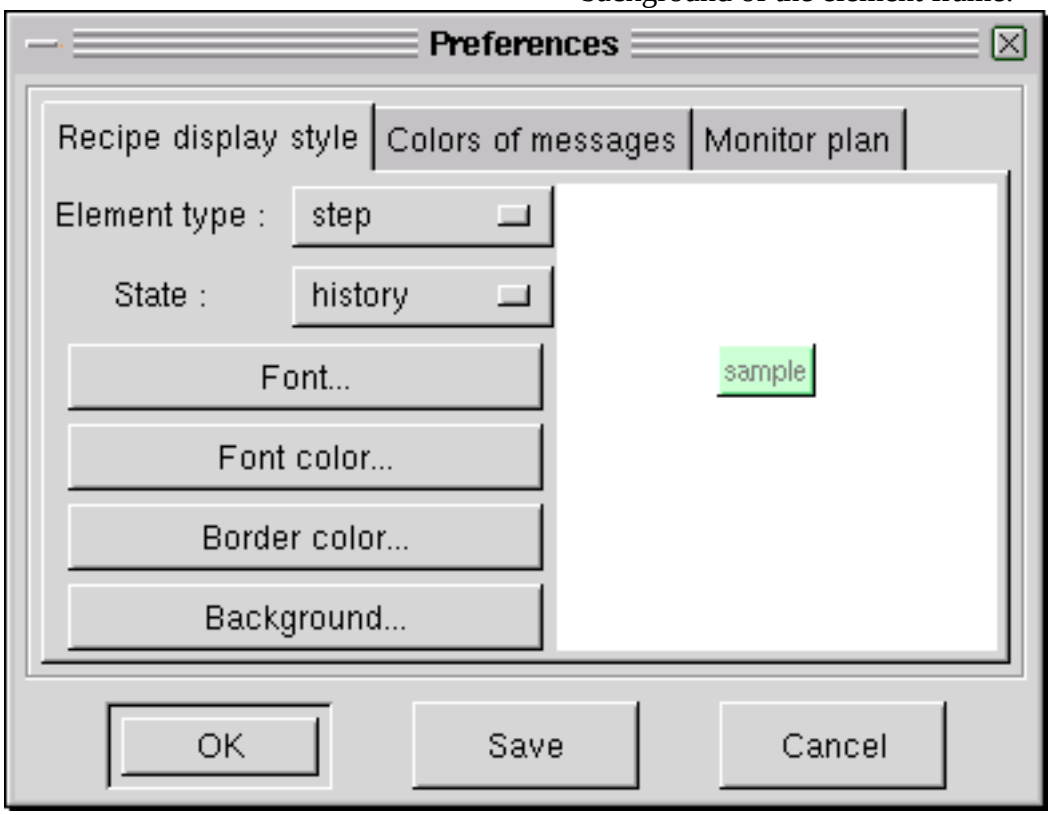

Figure 5.6. Edition of the recipe display preferences

5.8.2. Colors of messages

The Colors of message page can be used to the background colors used to display messages in Event and Scheduler Queue view. To select a new color, click on button. This will lauch a color selector.

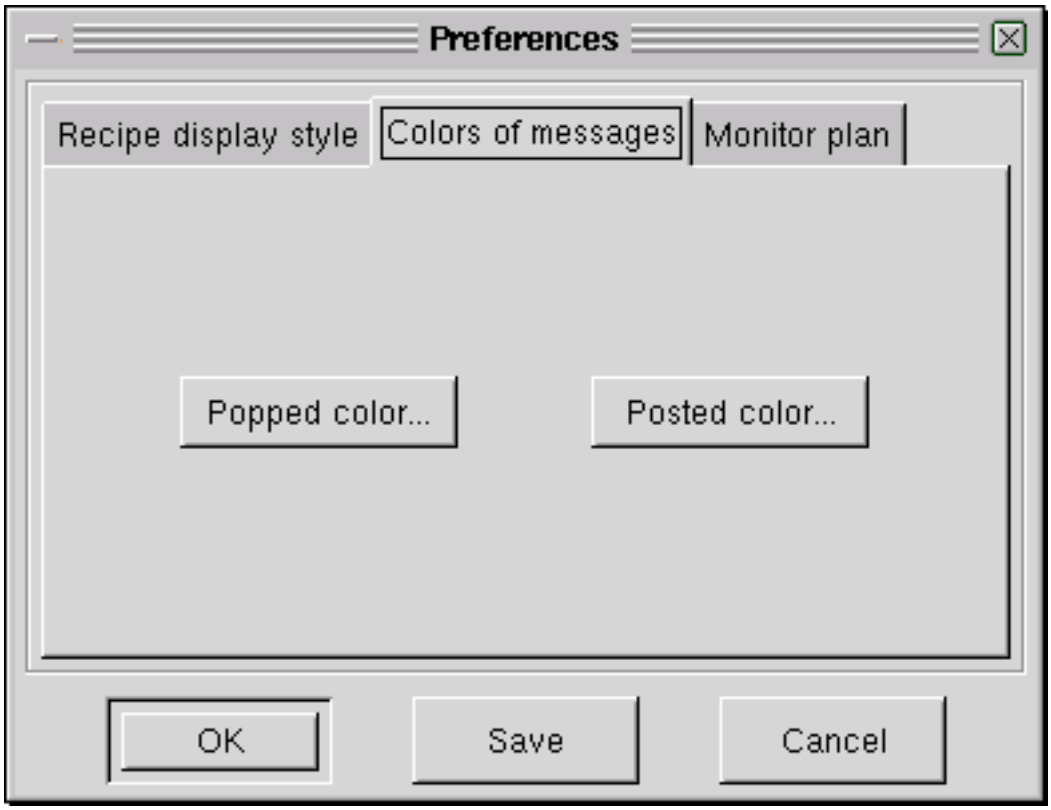

Figure 5.7. Edition of the message color preferences

5.8.3. Monitor plan

The Monitor plan page can be used to activate or disable the automatic destruction of plan windows when the execution of the plan is finished. This is a very useful feature if you have automonitoring turned on.

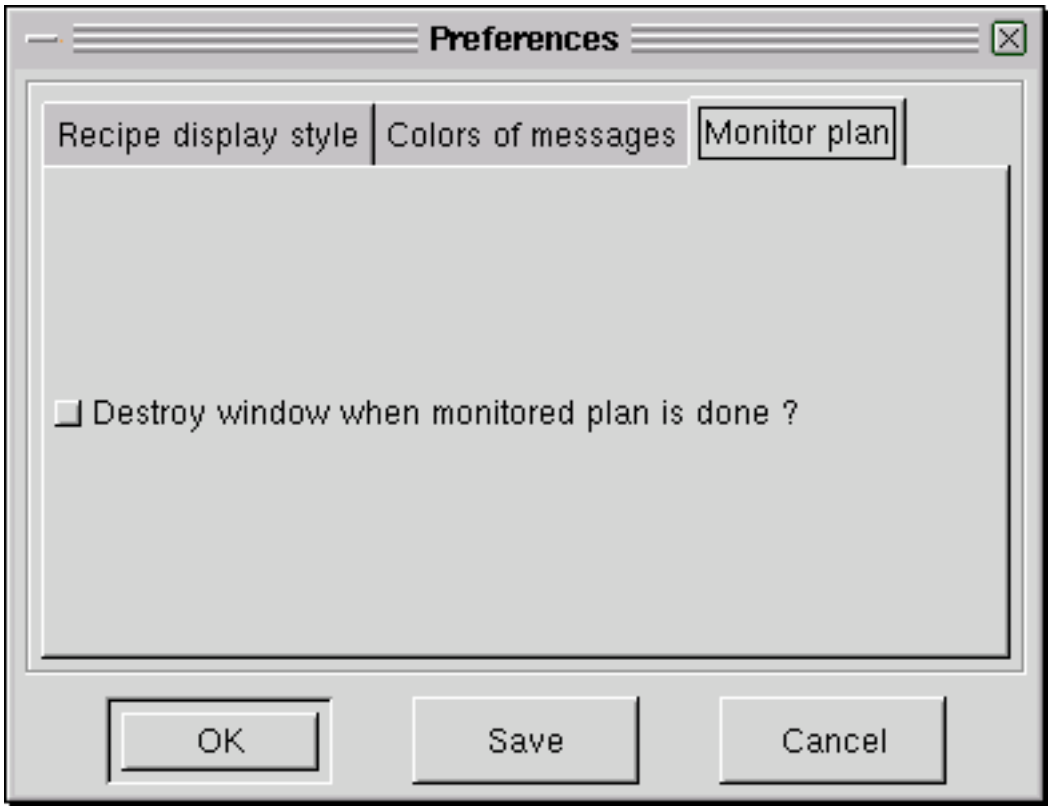

Figure 5.8. Edition of the plan monitoring preferences

Just click on Save button, to save your preferences.

5.9. Creating and editing recipes

5.9.1. Creating new recipes

To create a new recipe, you must specify a cookbook name in creating new recipe window and, of course, the recipe name. If you want the recipe to be in a completely new cookbook, just enter a new cookbook name in the cookbook field instead of selecting an existing one.

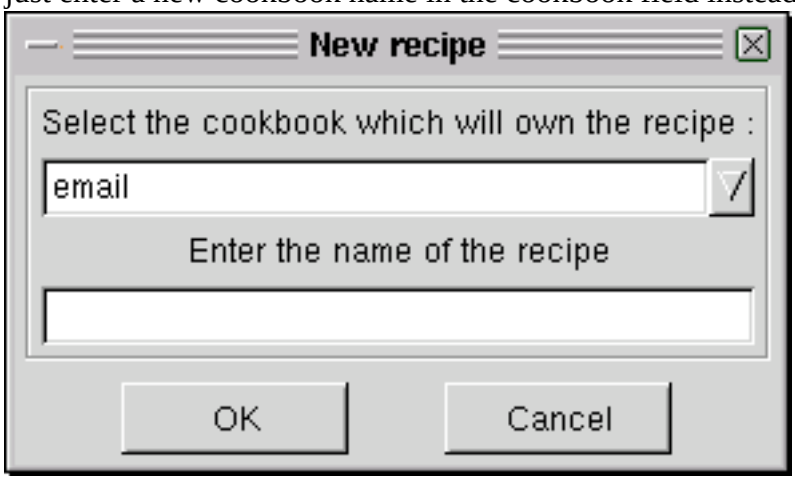

Figure 5.9. Recipe creation dialog window

5.9.2. Recipe edition window

To edit a recipe, you must select a recipe from the list given by the dialog window. You can also choose Edit from the popup menu that is displayed when you right-click on a recipe.

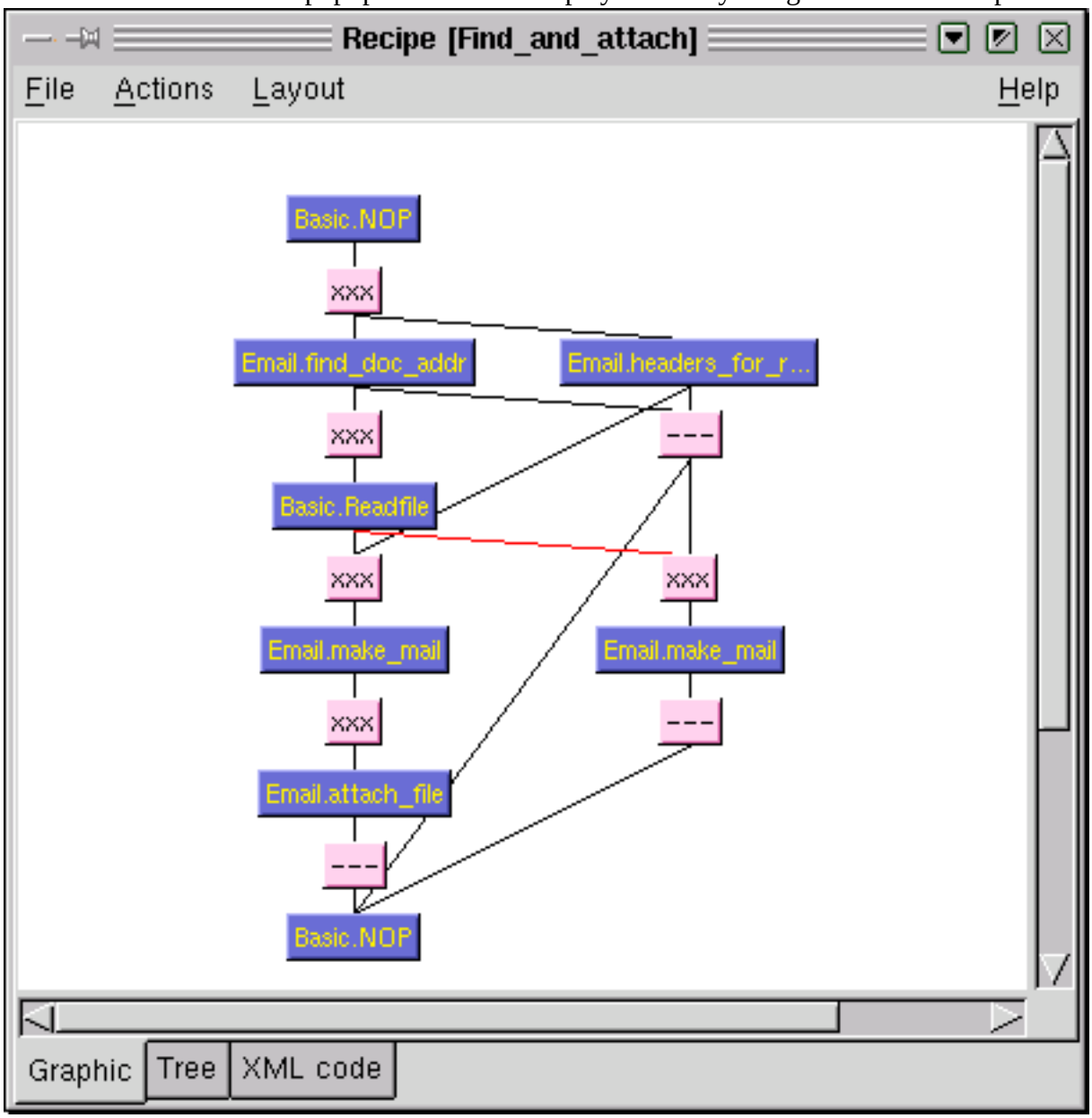

Figure 5.10. Recipe edition window

Clicking on the OK button launches the recipe edition window. This window provides 3 views of the recipe: a graphical view, a Tree view and an XML source view. Depending on the task you are achieving, using one view or the other will be easier.

5.9.2.1. The File menu

File->Save	Saves the recipe. In local mode, this saves the whole cookbook to disk. In connected mode, this replaces the recipe in Narval's memory with the edited recipe; if you want the modified recipe to be reloaded when Narval is restarted, you need to use the save recipes entry in the Narval menu of the main window.
File->Close	Close editing recipe window.

5.9.2.2. The Actions menu

Actions->Add step...	Displays a dialog window with a list of possible step targets. Recipes, actions and transformations are potential step targets.
Actions->Remove step...	Displays a dialog window with a list of recipe steps. Select the removing step.
Actions->Add transition...	Adds a transition to the recipe.
Actions->Remove transition...	Displays a dialog window with a list of recipe transitions. Select the removing transition.
Actions->Edit properties...	Displays a dialog window to configure the properties of the recipe.

5.9.2.3. The Layout menu

Layout->Simple Layout	Applies simple layout to graphical recipe view.
Layout->Layout 1	Applies layout 1 to graphical recipe view.
Layout->Layout 2	Applies layout 2 to graphical recipe view.

5.9.2.4. About the graphic view

On the graphic view, elements can be moved by dragging them with the left mouse button. Right clicking on an element pops up a local menu (see Figure 5.11.). Connecting two elements is very easy. Press SHIFT and click on first element, keep SHIFT pressed and click on second element. The connection is drawn on graphic view. If two steps were selected, a new transition is automatically added. Be careful when connecting a step and a transition since the selection order does matter.

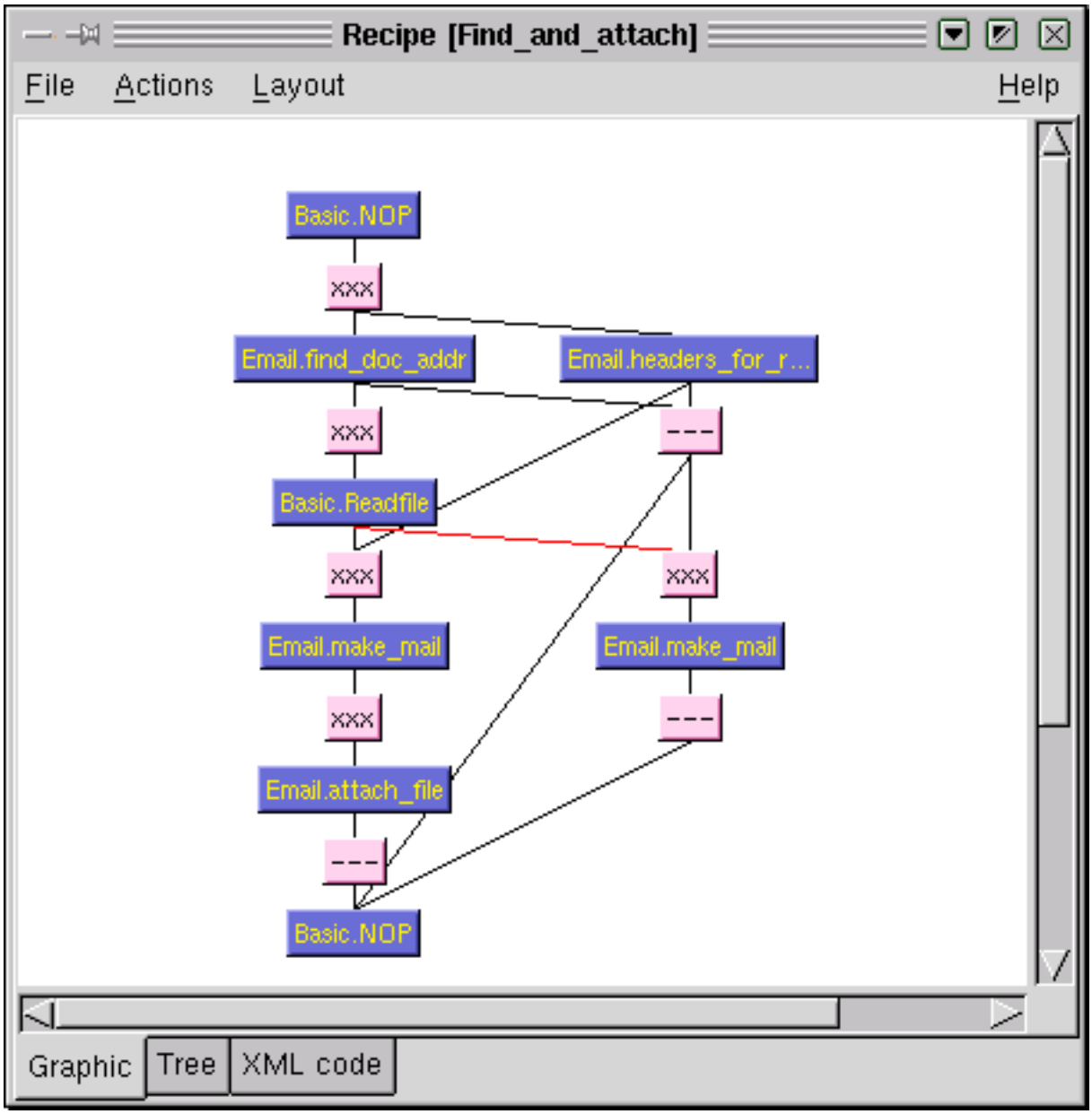

Figure 5.11. The recipe edition window, graphic view

5.9.2.5. About tree view

Tree view uses an XmlEditor widget to display the recipe. The XML editor displays XML nodes in a tree. Clicking with the mouse right button on a node pops up a menu. This menu can be used to add attributes, children or siblings to the node. The edition of attribute and text node value is done in the field under the tree. This field can be a single line text field, a drop down list or a multi line text field depending on the node being edited.

Sometimes, the allowed children for an element cannot be determined. This is the case for the arguments of a step, or XSL transformations. The editor detects thoses cases and enters a special mode, in which it is possible to enter some XML as text. Then by pressing Ctrl-Return, the XML is parsed and inserted in the tree.

Important

Do not forget to press Ctrl-Return, otherwise all the text you typed will be lost.

For more information on the meaning of the available elements and attributes, please refer to the Narval Technical Manual.

Right now, some elements of a recipe, such as transition conditions or step arguments, are not editable using the graphical editor. Right clicking on an element in the graphical view and selecting edit properties will bring up the tree view with the element selected.

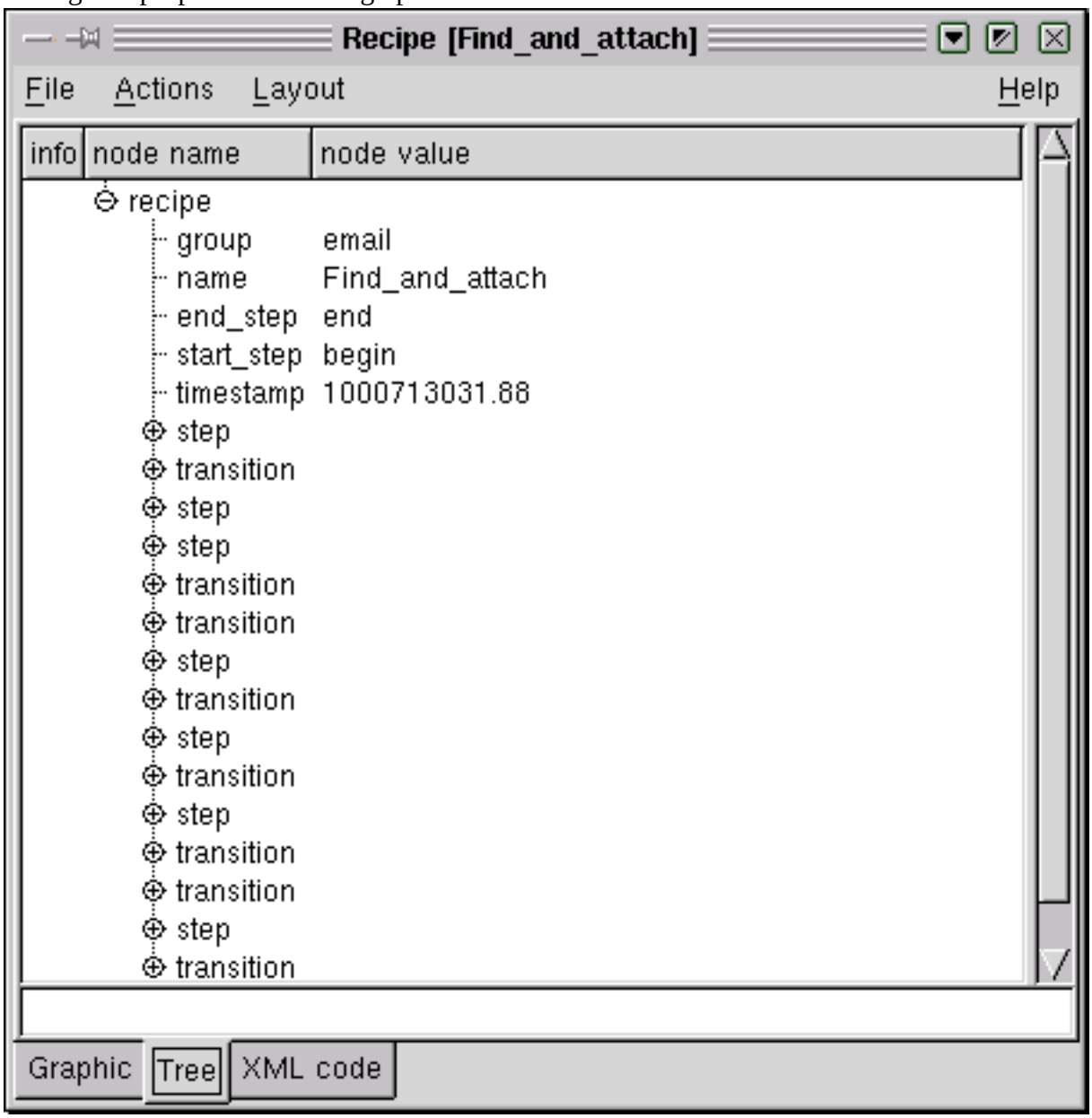

Figure 5.12. The recipe edition window, tree view

5.9.2.6. About XML Code view

XML Code view is a minimalist xml code editor. It is mainly intended for experiences Narval users, because using it expects that you have a good knowledge of the DTD for a recipe.

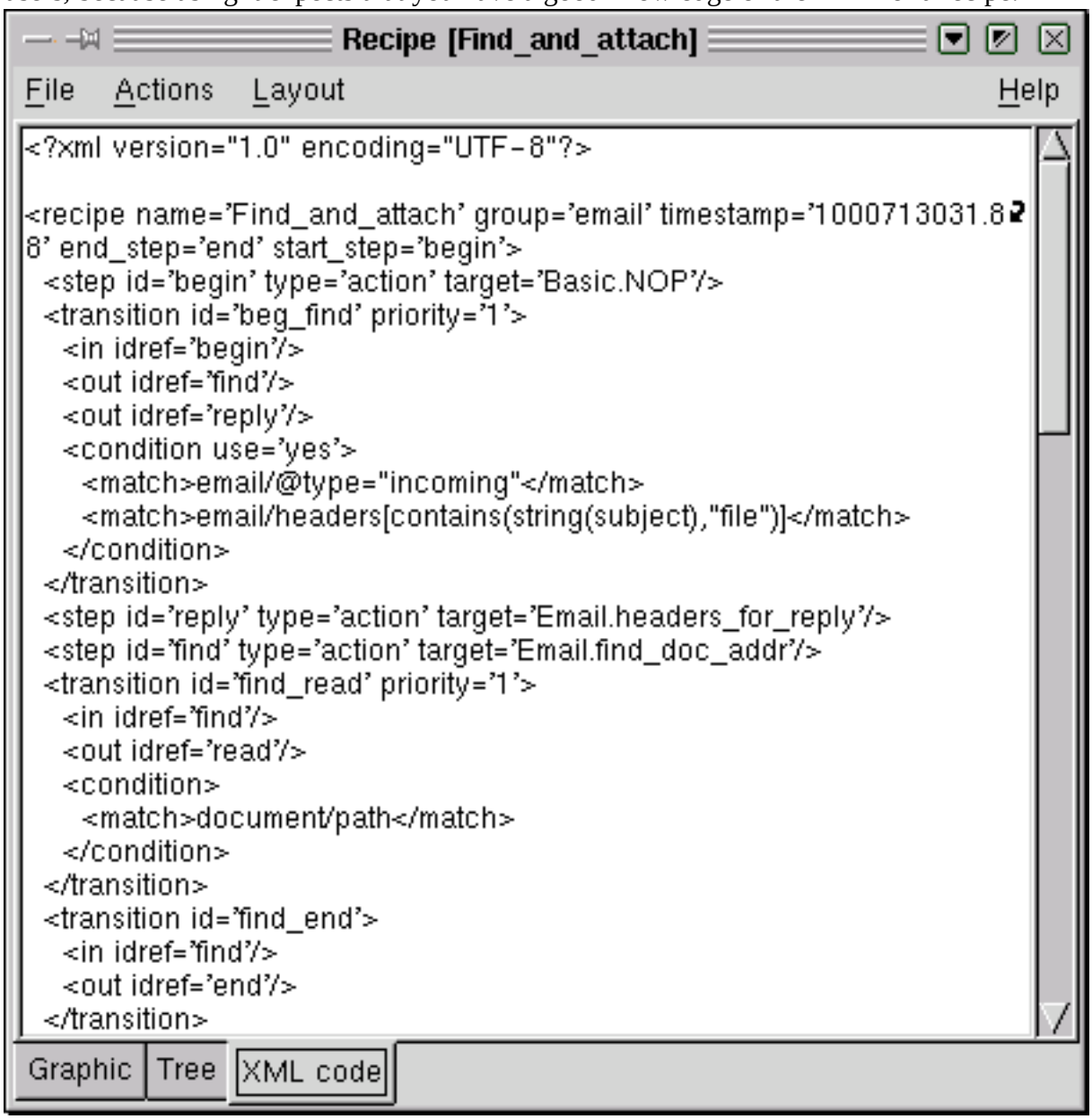

Figure 5.13. The recipe edition window, code view

5.9.3. Editing a recipe

The operations described hereafter are performed in local memory. To save to disk or to Narval's memory if you're working in connected mode, use File->Save (**Ctrl-S**). File->Close (**Ctrl-C**) closes the window without saving.

5.9.3.1. Changing the element layout

There are two ways of changing the layout of elements in a recipe. The first one is by clicking

on an element with the left button and dragging it to the desired position. The second one requires that the recipe be syntactically correct, which means that it must have a start step, an end step and that all steps but the start step have entries. It is in that case possible to use the Layout menu which offers (for now) three automatic layout methods.

> ## Remark
>
> We provide several methods because there is no, as far as we know, generic algorithm to layout graphs such as Narval recipes. Depending on the structure of the recipe, if it is wide or deep, if it has multiple crossed connections, the different methods will give different results. We suggest that you try all layouts on a new recipe to see which on fits the best.

5.9.3.2. Adding a step

It is possible to choose an action, a transformation or a recipe in the memory window and to drag it with the mouse left button to the recipe edition window. This will add a new step with the corresponding element as target.

An alternative method is using the Actions->Add step which brings up an element selector, or add step in the pop up menu that is displayed by right clicking in the graphical view. On windows, this is the only way of adding a new step, since drag and drop won't work.

5.9.3.3. Adding a transition

There are several ways to add a transition: first, Actions->Add Transition. In that case, the new transition in inserted in the middle of the recipe, which can make it difficult to find it.

The second method is to use a popup menu. By pressing the right button on the background of the edit window, a menu pops up. Selecting add transition, inserts a transition in the recipe, at the point that was clicked.

The last method is connecting two steps together (see next section). A new transition is automatically created to make the connection possible.

5.9.3.4. Connecting elements

Creation of execution paths in a recipe is done by connecting elements together. It is possible to connect a step to a transition or a transition to a step. A connection is established by clicking on the start and end elements while holding the **Shift** key pressed.

> ## Tip
>
> It is also possible to connect a step on another step. This does not connect both steps directly, but rather adds a new transition in between in the first place, and sets connections properly afterwards. Connecting a transition to another transition has no effect.

5.9.3.5. Changing the properties of a recipe

By using Recipe->Properties, or the menu that pops up by clicking with the mouse right but-

ton on the background of the edit window and choosing edit properties, the recipe properties edit dialog appears. Several fields are available in that dialog:

Name	the name of the recipe
Start step	The start step of the recipe. A recipe must have a start step, otherwise Narval will not be able to transform it into a plan. This field allows selection of the start step from a list of all the steps in the recipe.
End step	The end step of the recipe. A recipe must have an end step, otherwise Narval will not be able to transform it into a plan. This field allows selection of the end step from a list of all the steps in the recipe.
Restart mode	By changing the state of the button, it is possible to change the behaviour of Narval regarding plans instanciated from this recipe.By default, the option is off, and plans are terminated when they reach their end step. When the option is on, the plan restarts from the beginning after the end step is over.
Decay	This parameter gives the lifespan of a plan and associated memory elements, in seconds. If unspecified, a default value of 90seconds (one minute and a half)is used.

5.10. Using the command line interface

Horn can be used as a graphical client or a command line client. The latter is especially useful when writing scripts or when integrating Narval with another application.

Once you have launched Horn with the --cli command line flag, you get a command line prompt that you can use to input commands. If you prefer, you can also execute commands from a script file, by using the --exec option.

5.10.1. Available commands

Commands in Horn are grouped into seven topics.

5.10.1.1. Package commands

- install <name.npm>

  installs a package in the current `NARVAL_HOME`
- uninstall <name.manifest>

installs a package in the current `NARVAL_HOME`

5.10.1.2. Narval commands

- connect_to <host>:<port>

  connects to a narval running on host and listening on port
- disconnect

  disconnect from narval
- local

  work in local mode. Load data from current `NARVAL_HOME`

5.10.1.3. Memory commands

- get_memory

  loads the memory from the remote Narval server. This can be a lengthy operation.
- select_elements <xpath>

  print all the elements from Narval's memory matching the XPath. This requires loading all the Memory.
- get_element <id>

  print the XML code of the element with the specified identifier
- add_element <xml>

  create a new element in Narval's memory, with the specified XML as content
- remove_element <id>

  removes the element with the specified identifier
- replace_element <id> <xml>

  replacef the element with the specified identifier with the new element
- start_plan <group>.<name>

  instanciate a new plan from the specified recipe and run it
- save_recipes

  save all the recipes in Narval''s memory
- save_transforms

  save all the transformations in Narval's memory

5.10.1.4. Debug commands

- debug_suspend

  suspends the interpreter
- debug_step_one_step

  let the interpreter perform an elementary operation.
- debug_continue

  resume normal interpreter operations

5.10.1.5. Recipe commands

- new_recipe <name> <cookbook>

  create a new recipe in the specified cookbook
- save_recipe <group.recipe>

  commit changes to the specified recipe
- add_action_to_recipe <group.recipe> <module.action> [id]

  adds an action to the recipe, with an optional identifier.
- add_transform_to_recipe <group.recipe> <group.transform> [id]

  adds an transformation to the recipe, with an optional identifier.
- add_recipe_to_recipe <group.recipe> <cookbook.recipe> [id]

  adds an recipe to the recipe, with an optional identifier.
- add_transition_to_recipe <group.recipe> [id]

  adds an transition to the recipe, with an optional identifier.
- remove_element_from_recipe <group.recipe> <id>

  remove a step or a transition from the recipe
- remove_connection_from_recipe <group.recipe> <id1> <id2>

  disconnect a step and a transition
- connect_elements <group.recipe> <id1> <id2>

  connect a step and a transition
- add_argument_to_step <group.recipe> <id> <xml>

  add an argument to a step
- add_condition_to_transition <group.recipe> <id> <xml>

  add a condition to a transition

- add_attribute_to_recipe <group.recipe> <name> <value>

  add an attribute to the recipe

5.10.1.6. Logging commands

- put_into_logfile

  log Narval's messages to a file
- close_logfile

  stop logging

5.10.1.7. Other commands

- help [cmd]

  print a help message about the command or topic
- quit

  quits horn

Chapter 6. Narval Installation

This chapter introduces a few notions on the installation of Narval, more specifically, what is needed, and what goes where. We suggest that you read the readme file which came with your distribution of Narval for more information. The file for the release corresponding to this version of the documentation is reproduced in Appendix A..

6.1. Prerequisites

Narval has been written using as much existing applications as possible; that's why its installation needs preliminary installation of:

- Python 2.1 (Narval programming language),
- PyGTK >= 0.6 (Python graphical library used by Horn, Narval graphical interface),
- PyXML >= 0.6.6 (Python DOM implementation used by Narval).
- 4Suite >= 0.11.1 (Python XSLT and XPath implementation used by Narval).
- xmlrpclib >= 0.9.9 (Python XMLRPC implementation used for client-server communication between Narval and Horn).

On windows platforms, you only need Python, since the Narval installer includes all the required files.

These several applications can be downloaded from the following URLs:

Python 2.1	http://www.python.org/2.1/ [http://www.python.org/2.1/]
PyGTK 0.6.X	http://www.gtk.org/ [http://www.gtk.org/]
PyXML-0.6.6	http://sourceforge.net/project/showfiles.php?group_id=6473 [http://sourceforge.net/project/showfiles.php?group_id=6473]
4Suite 0.11.1	http://www.4Suite.org [http://www.4Suite.org/]
xmlrpc 0.9.9	http://www.pythonware.com/products/xmlrpc/ [http://www.pythonware.com/products/xmlrpc/]

Table 6.0. URLs of the applications to download

6.2. Programs

On Unix systems, installing Narval installs two applications:

- `narval` is the application core and contains memory and executes the plans and handles the elements (see Chapter 3., *Narval generic behaviour*).
- `horn` is Narval graphical interface which provides various ways for controlling and visualizing the plan execution.

`narval` can be run without `horn` but it doesn't have any graphical interface and provides very little information on its internal state. Anyway, it can be run with the `--debug` option, which makes it more verbose.

For `horn` use, see Chapter 5., *Horn, Narval user interface*.

During installation, they are put in the following directories (unless you have a very specific system configuration):

- `narval` in `/usr/bin/`,
- `horn` in `/usr/bin/`.

On Windows systems, the installer creates two batch files that can be used to launch Narval (`narval.bat`) and Horn (`horn.bat`). These two files are set in `C:\Program files\Narval-1.x`. Additionnaly, two shortcuts are created in the start menu for these batch files. Please note that the shortcut that lauches narval does it with the --listen-on 8787 command line option.

6.3. Data

The Narval application uses numerous data: DTDs[1], recipes, modules, transformations, logos and a memory initialisation file. This data are organized as follow:

Directory	Content
`dtd/`	contains the various DTDs used by Narval; in particular, `Narval.dtd` contains the DTD dealing with Narval core (recipes, plans, steps, transitions, actions and so on)
`graphics/`	contains various graphics used by Horn
`recipes/`	contains the files in which recipes are described (all the files in this directory are au-

[1] **Document Type Definition (DTD):** Description of the content of an XML file, particularly the elements and attributes it might contain.

Directory	Content
	tomatically loaded in memory during `narval` initializing)
`modules/`	contains the the python modules providing the actions(all the files in this directory are automatically loaded in memory during `narval` initializing)
`transforms/`	contains several subdirectories, each holding several XSL transformations (grouped by theme). All the transformatoins in these directories are automatically loaded in memory during `narval` initializing)
`doc/`	contains several subdirectories, in which the documentation for the Narval applications you install will be set.
`data/`	contains especially the file `memory.xml` used for memory initialisation when running `narval`. Most recipes that produce result in files, or require configuration files will use this directory to store the files
`extensions/`	contains miscelaneous python extensions that can be required by python modules to work correctly.

Table 6.1. Content of the data directories

These data are stored in two different places. On Unix systems:

- in `/usr/share/Narval/` (or the equivalent in your system) where they are shared by all the users,
- in `~/.narval/` (with ~ representing user home directory) where they belong to a specific user.

On Windows system:

- in `C:\Program Files\Narval-1.x\share` (or the equivalent in your system) where they are shared by all the users,
- in `appdata\narval_data` (with `appdata` is the application data folder as read from windows registry.

Only the data in `.narval/` are used by Narval and Horn. This directory is created from

`/usr/share/Narval/` content when `narval` or `horn` is executed for the first time.

6.4. Code

Narval and Horn source code is installed in `/usr/lib/python2.1/site-packages/narval/` (`/usr/lib/python2.1/` is Python library directory and might be different in your system. On Windows systems, the default is `c:\Python21\narval`).

Glossary

action	Conceptually elementary transformation Narval applies on elements.
element	Applicative computing entity located in memory that might be handled by an action. An email, a Web page, a plan or a recipe are elements.
step	Basic brick of a recipe that can be an action.
memory	Storage place of the elements in which they can be accessed by actions.
module	Container of conceptually close actions.
plan	Instance of a recipe allowing its execution.
recipe	Sequence of steps linked by transitions describing a functionnality of Narval.
transition	Link between a set of origin steps and a set of destination steps that can have a condition on elements found in memory.

Appendix A. The README file for the current release

A.1. Unix systems

```
README for Narval version 1.1
$Date: 2001/10/13 15:42:48 $ $Revision: 1.10 $
DOWNLOADING
-----------
To run Narval, you need the following packages on your machine:
 * python 2.1 available from http://www.python.org/2.1.1/ The good old
   Python interpreter.
 * PyXML 0.6.6 or above available from
   http://sourceforge.net/project/showfiles.php?group_id=6473
 * 4Suite 0.11.1 or above available from http://www.4suite.org.
   4Suite is a DOM/XSL/XPATH implementation from Fourthought, Inc.
 * PyGTK version 0.6 or above available from
   http://www.daa.com.au/~james/pygtk/ and http://www.gtk.org/.
   Python bindings for the GTK gui toolkit.
 * xmlrpc version 0.9.9 or above available from
   http://www.pythonware.com/products/xmlrpc/
 * xmltools version 1.3 or above available from http://www.logilab.org
And of course, Narval itself, availalble from http://www.logilab.org/n
INSTALLING
----------
1. site-wide installation
-------------------------
Of course, you'll need to be root for this.
Untar the packages you downloaded in a directory using:
# tar xzf Narval-1.1.tar.gz
then
# cd Narval-1.1
Now use the setup.py script that's in the tools directory. In order to
your local settings, you may want to edit the PREFIX variable in the
"defines"
section at the top of the script.
# python setup.py install
2. user installation
--------------------
To use Narval without installing it for every user of your system, jus
the packages you downloaded in a directory using:
$ tar xzf narval-1.1.tar.gz
You'll get a Narval-1.1 directory that you'll add to your PYTHONPATH e
variable:
$ export PYTHONPATH=$PYTHONPATH:~/Narval-1.1
LAUNCHING
```

```
---------
1. after site-wide installation
-------------------------------
If Narval was set up using setup.py or a RPM package, you should have
symbolic links somewhere in your path (probably /usr/bin), called 'nar
'horn' and 'npm' which you can use to launch the application.
2. after user installation
--------------------------
If you have not used the setup.sh script, you must set the NARVAL_SHAR
environment variable to point to the share subdirectory of the place w
unpacked Narval. For instance assuming it was in your home directory,
use:
$ export NARVAL_SHARE=~/Narval-1.1/share
with an sh-compatible shell, or:
$ setenv NARVAL_SHARE ~/Narval-1.1/share
with a csh-compatible shell.
Then you can use
$ python Narval-1.1/narval/Engine.py
or
$ python Narval-1.1/narval/Horn.py
USING
-----
1. quick start
--------------
To launch Narval with http connection and remote monitoring enabled, u
'--socket-manager --listen-on 8787' command line options.
You will then be able to use the "Narval -> Connect" menu from Horn to
to Narval (using your hostname and 8787 as the port). Fool around with
if you want, but don't lose too much of your time and...
Read the tutorial at http://www.logilab.org/narval/doc.html that descr
to download, set up and start an existing application available from
ftp://ftp.logilab.org/pub/narval/applications...
But be aware that you most probably won't get away without reading the
manuals :-).
2. going further
----------------
Read the tutorial first (see http://www.logilab.org/narval/doc.html).
It describes what you can do with Narval, and shows how to build a rec
lauch it from Horn, step by step. After that, reading the user manual
provide useful information. Feel free to subscribe to our mailing list
(http://www.logilab.org/narval/dev.html#ml) and questions there, we sh
glad to answer them.
          Have Fun
          The Logilab Team
```

A.2. Windows systems

```
README for Narval version 1.1 on Windows systems
$Date: 2001/10/13 15:42:48 $ $Revision: 1.9 $
DOWNLOADING
-----------
To run Narval, you need the following packages on your machine:
```

```
 * python 2.1 available from http://www.python.org/2.1.1/. The good ol
   Python interpreter.
 * Narval itself, available from ftp://ftp.logilab.org/pub/narval/
INSTALLING NARVAL
-----------------
Run all the installers in the following order :
 * python (if python is not installed on your system)
 * narval
Running the Narval installer will offer to install several other packa
you. Unless you are sure that a compatible version of each of these pa
already installed on your machine, it is strongly advised that you use
'full' setup.
The Narval installer has been tested with Windows 98/2K. We are intere
in hearing from you about experience with that installer, so that we c
enhance the installation process.
LAUNCHING
---------
You can now use the shortcuts in the start menu to launch Horn or Narv
USING
-----
1. quick start
--------------
To launch Narval with http connection and remote monitoring enabled, u
--socket-manager --listen-on 8787 command line options. This is what y
when using the Start menu shortcut.
You will then be able to use the "Narval -> Connect" menu from Horn to
to Narval (using your hostname and 8787 as the port). Fool around with
if you want, but don't lose too much of your time and...
Read the tutorial at http://www.logilab.org/narval/doc.html that descr
to download, set up and start an existing application available from
ftp://ftp.logilab.org/pub/narval/applications...
But be aware that you most probably won't get away without reading the
manuals :-).
2. going further
----------------
Read the tutorial first (see http://www.logilab.org/narval/doc.html).
It describes what you can do with Narval, and shows how to build a rec
lauch it from Horn, step by step. After that, reading the user manual
provide useful information. Feel free to subscribe to our mailing list
(http://www.logilab.org/narval/dev.html#ml) and questions there, we sh
glad to answer them.
          Have Fun
          The Logilab Team
```

Appendix B. Credits

B.1. credits

Evelyn Mitchell helped proof-read the 2001/05/23 version.