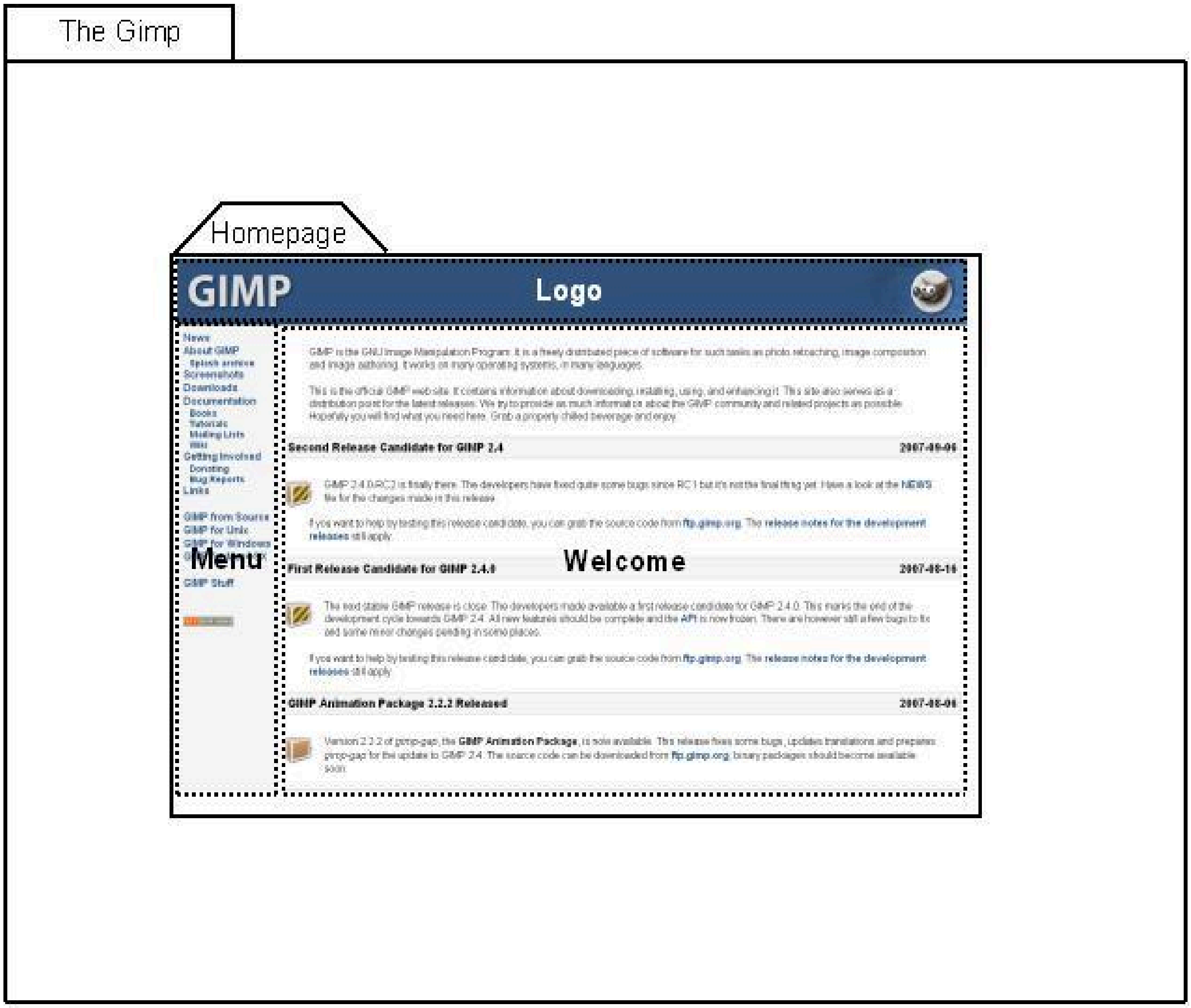
Howto use the List element

In this HowTo we use two examples to explain how to use the U*ia*ML List element.

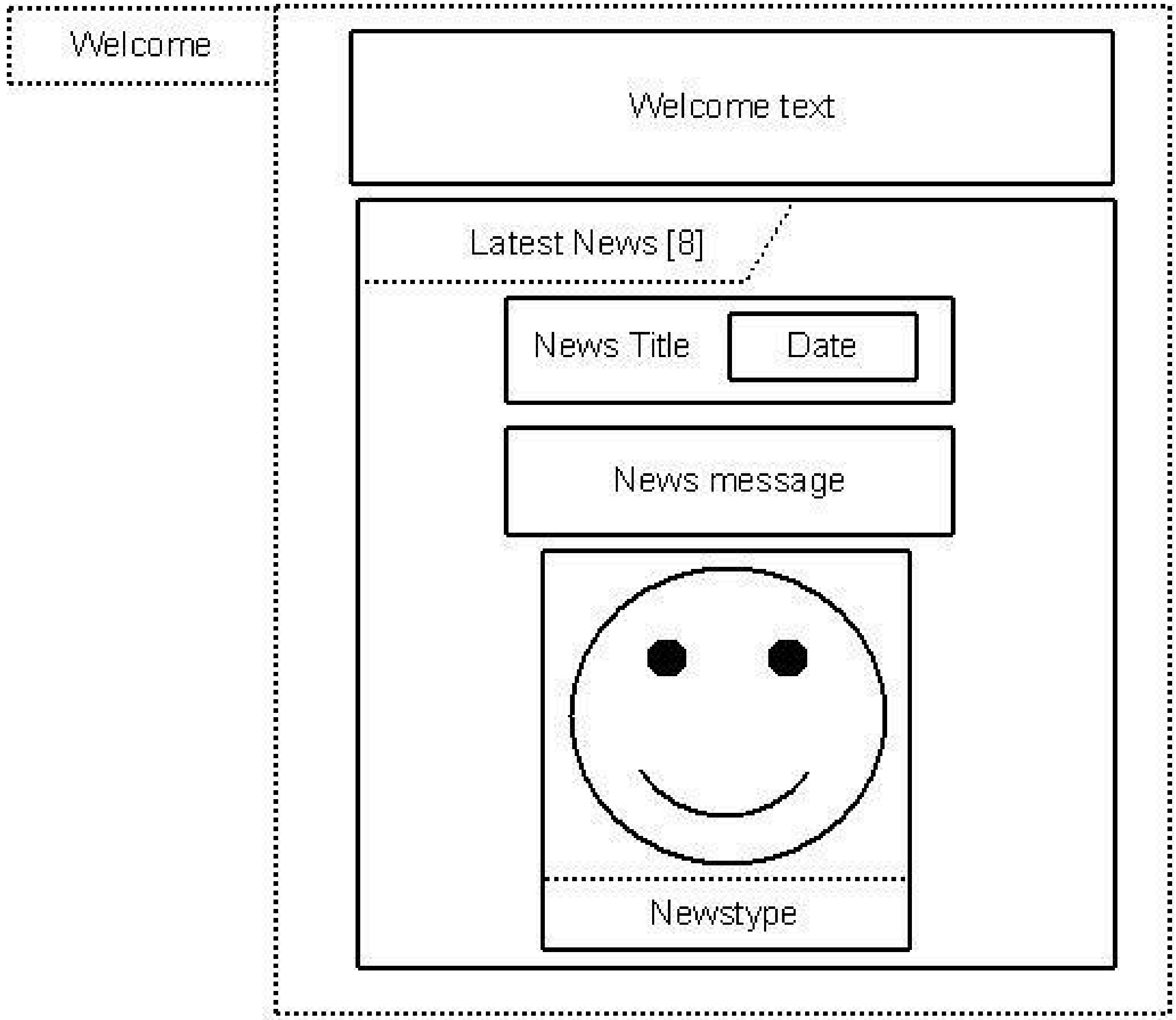
Example 1: The news items

This example is based on the homepage of the GIMP (http://www.gimp.org). The homepage of the GIMP can be dissected into three contentareas (Logo, Menu and Welcome). See the figure below. Note that the screenshot of the website is not meant to be present within a U*ia*ML documentation. But since we are reverse engineering, it becomes clearer which contentarea covers which part of the homepage.



Since this howto is about the List element we will not focus on the Welcome contentarea.
The welcome content area consists of an introduction text followed by the latest news of the gimp project.
The latest news is a **list** of news items. Modelling a single news item and put it into a U*ia*ML List element is enough to express the list of news items, since each news item has a title, a date, an image that represents the type of the news

and some text that tell us more about the news.
All that is left is to complete this example is write down the propertie fields of each contentarea element (CAE) within the “Welcome” contentarea.



TextLabelID	Welcome text
Intended content	Some introduction about the gimp and this site.

ListID	Latest News
Nr of items	8
Sort element	CAE:News Title . CAE: Date

Sort order	Alphabetical descending
Recursion	FALSE

TextLabelID	News Title [n]
Intended content	Title of the news item

TextLabelID	Date [n]
Intended content	Publish date of the news item

TextLabelID	News message [n]
Intended content	Text that tells more about the news

ImageID	Newstype [n]
Intended content	Image to express the type of the news

Only the latest 8 news items are published on the site, so the number of items property of the “Latest News” list element is set to 8. The news items are meant to be listed by date. But since the CAE “Date” is a subelement of “News Title” the relative path toward the text element “Date” from the list element “Latest News” is given. Notice that the notation between two U*ia*ML elements requires a dot as separator (the dot has been chosen, so that the U*ia*ML notation shows some similarity with the Object Oriented programming notations).
The sort order of the news is descending so that the latest news will be on top and because there is no recursion involved we set the recursion property to false (more about the recursion property in the second example).
Another thing you might notice is the “[n]” behind the CAE's of the list element.
Without that notation we must have written:

TextLabelID	News Title [1]
Intended content	Title of the news item

TextLabelID	News Title [2]
Intended content	Title of the news item

TextLabelID	News Title [3]
Intended content	Title of the news item

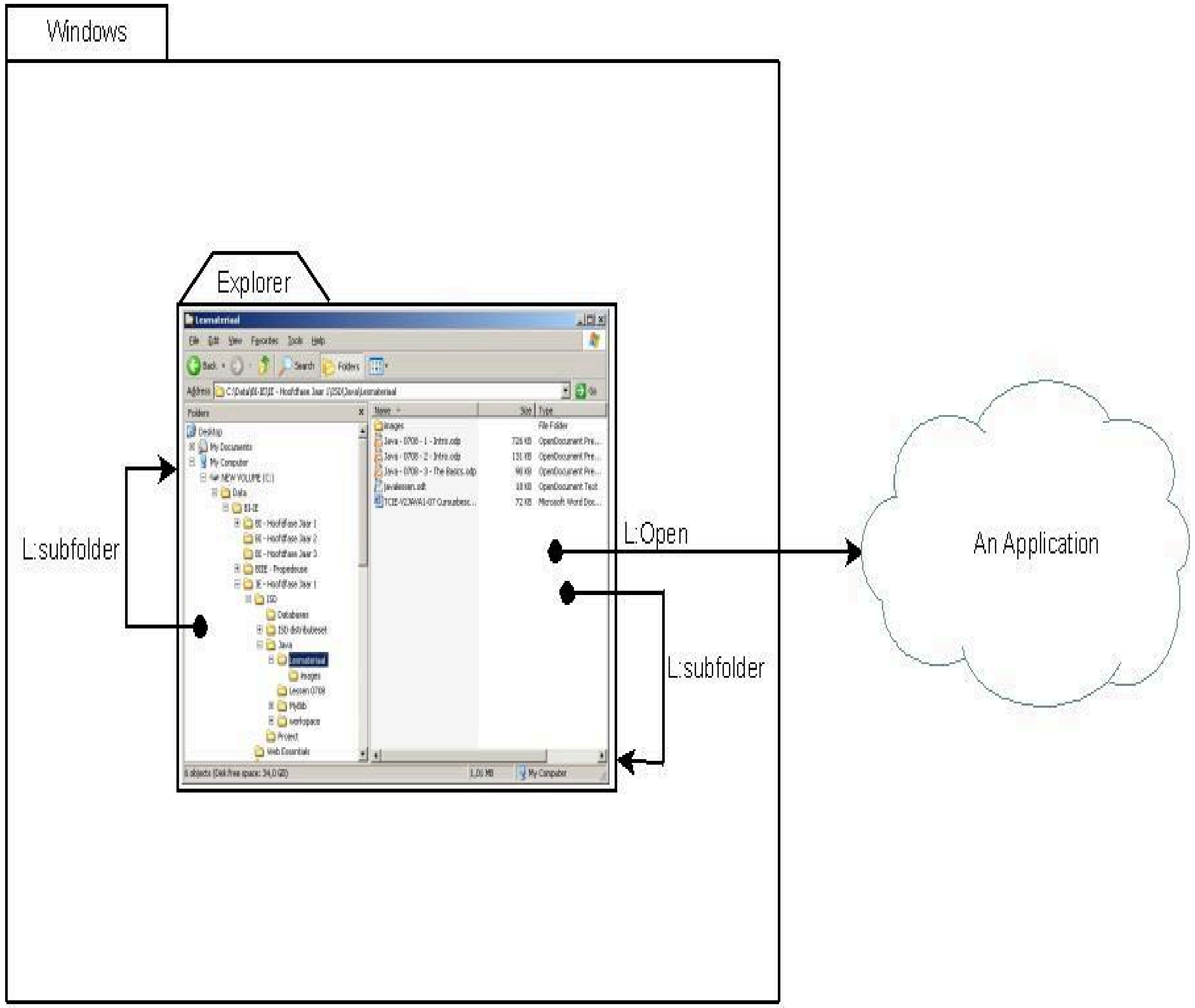
......

TextLabelID	News Title [8]
Intended content	Title of the news item

And had to repeat this for all the other CAE's of the list element. But because the property remains the same with each instance of the CAE the short form “[n]” is sufficient. Which doesn't mean that the long form never occurs. In general the short form will occur with dynamically filled list elements. As soon as there is some static content involved the long form might be required.

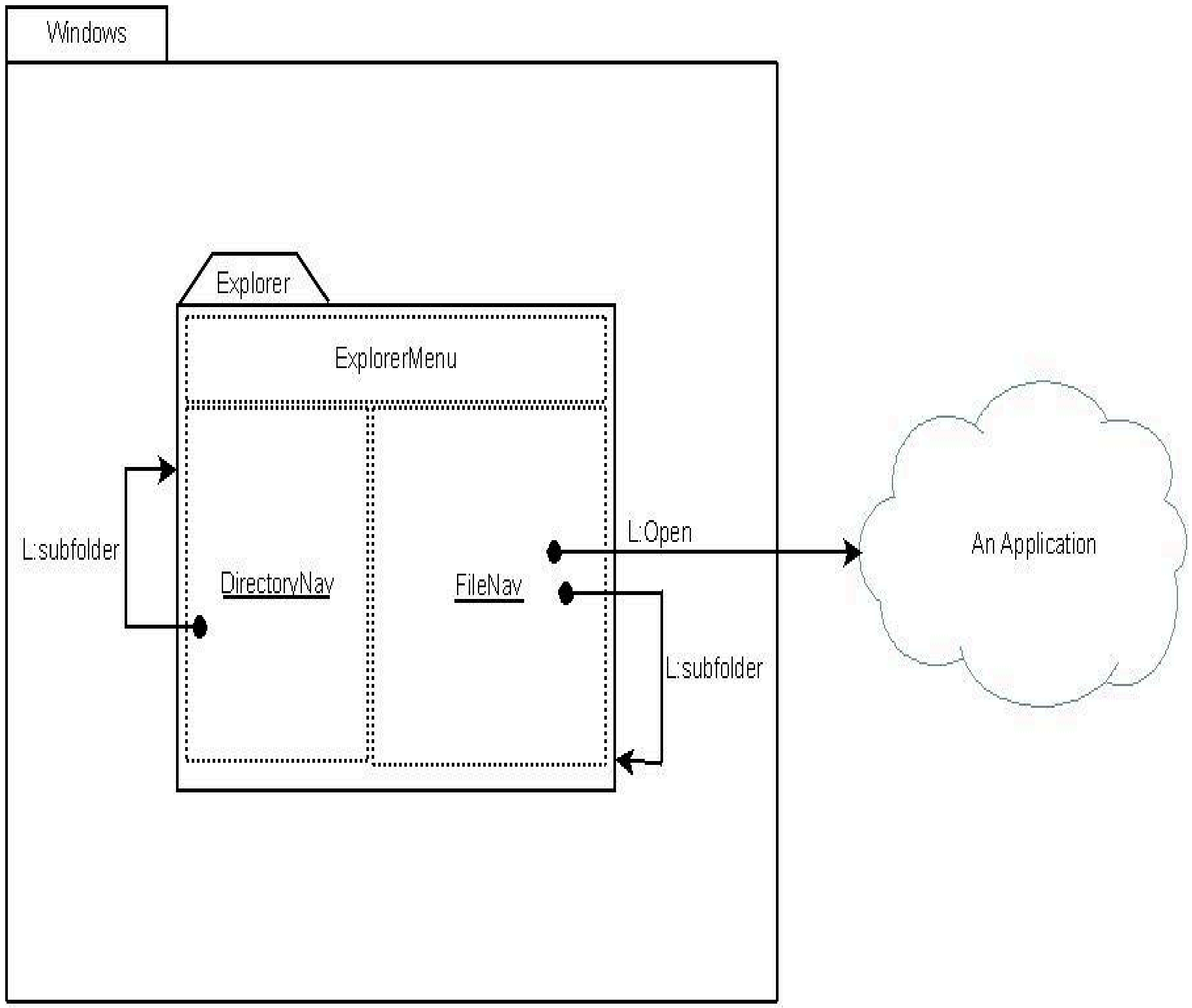
Example 2: The file explorer

The file explorer is another example that makes use of the list element. Again we are about to reverse engineer a model. So again we use a screenshot (naughty naughty) to kick start this example.



To simplify this example we only focus on the directory tree on the left and the directory content on the right. Which

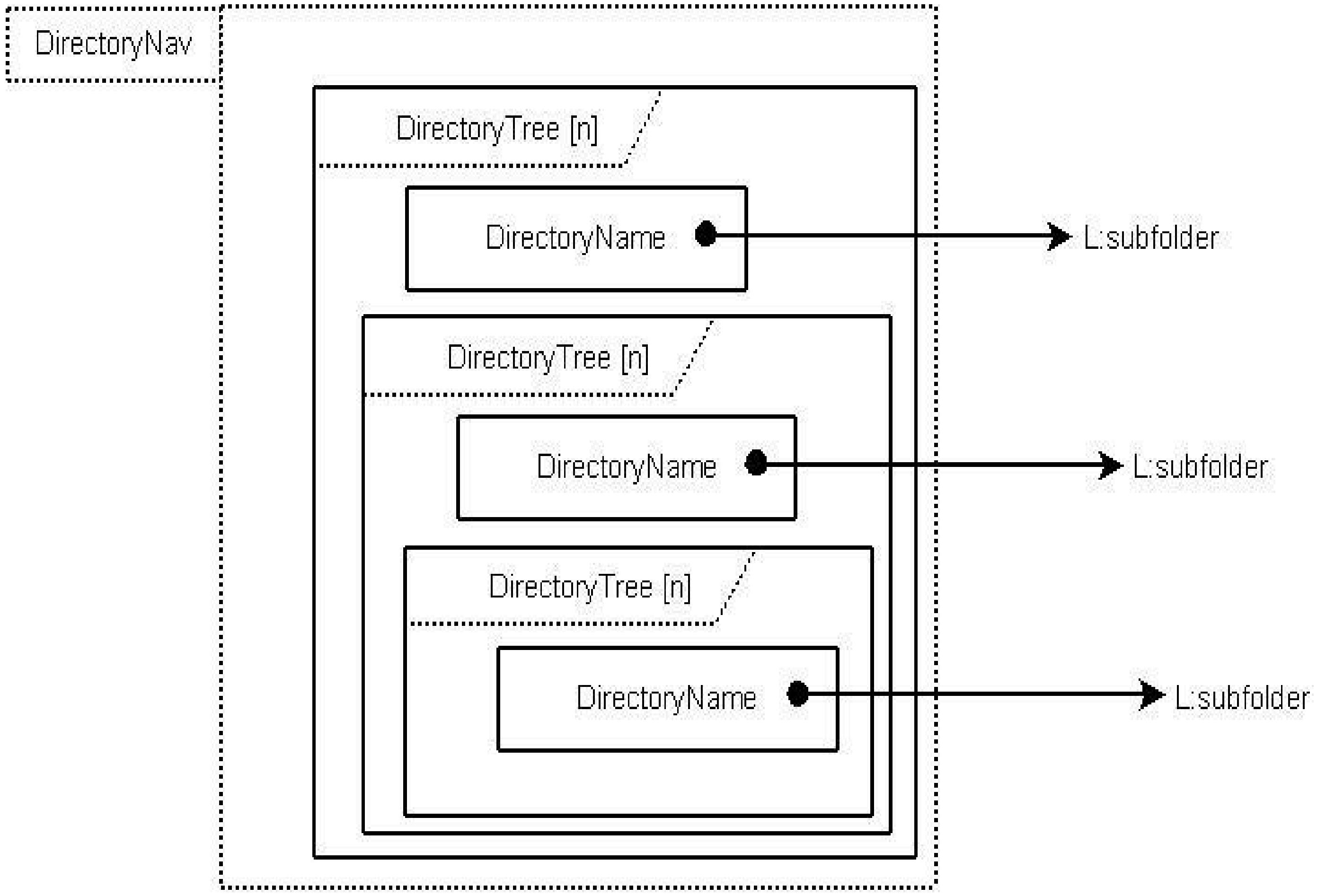
are actually to contentarea that we will refer to as “DirectoryNav” and “FileNav”. The “DirectoryNav” contentarea allows us to navigate between subdirectories. The “FileNav” contentarea offers us the possibility to open subdirectories and to open an associated application.
Because it isn't possible to predict which application will be opened when a file gets opened, we use a cloud symbol outside the U*ia*ML site symbol to represent this.



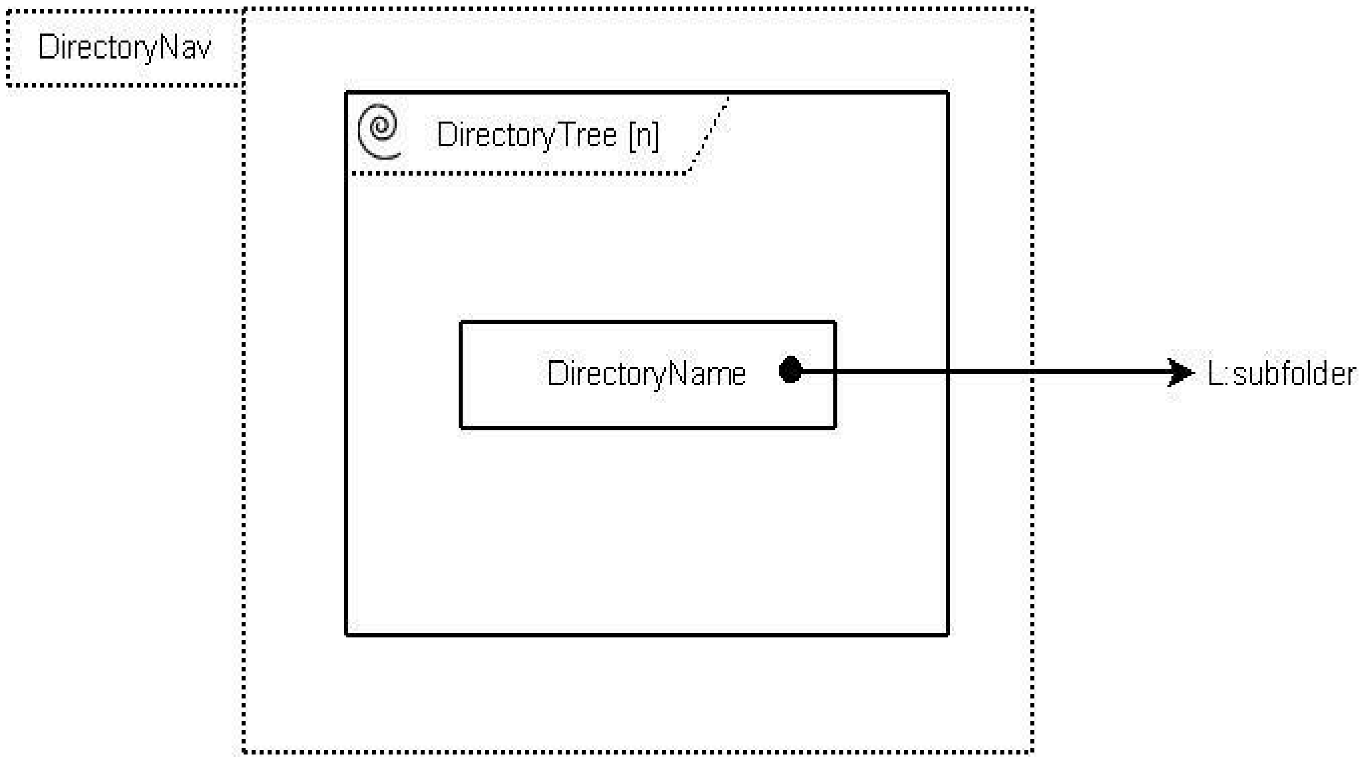
Contentarea ID	DirectoryNav
Description	The “DirectoryNav” contentarea allows us to navigate between subdirectories.

Contentarea ID	FileNav
Description	The “FileNav” contentarea offers us the possibility to open subdirectories and to open an associated appliacation.

Now let's take a closer look at the contentarea views (CAV).
The “DirectoryNav” is actually a list of directory names. Opening a directory name leads to an other list of directory names. So it's a list in a list. Modelling it like in the figure below is no solution.



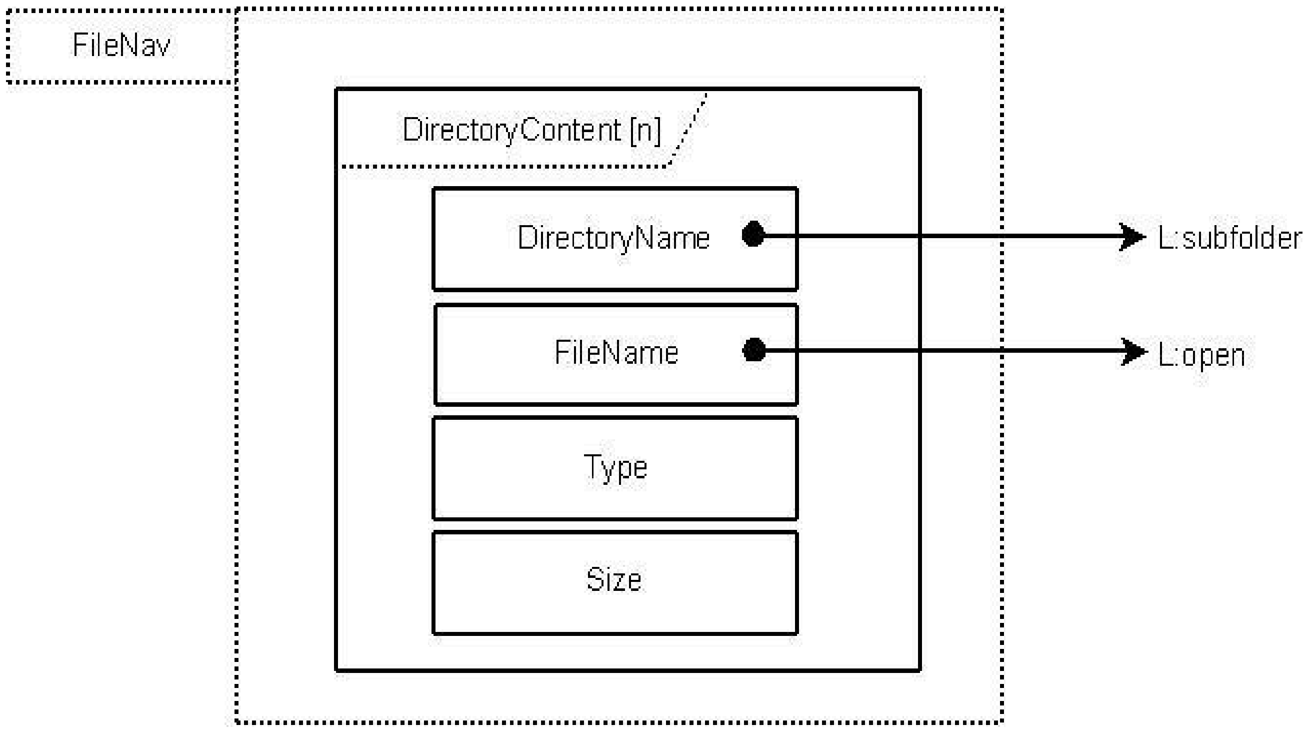
The problem we encounter is that we don't know how many subdirectories there are. So we don't know how many levels deep we need to model the DirectoryTree list. But what you can see from the model above, is that the DirectoryTree list element is recursive. The U*ia*ML list element allows to express this by drawing a spiral in the upper left corner of the list element.



ListID	DirectoryTree
Nr of items	N
Sort Element	CAE:DirectoryName
Sort Order	Alphabetical ascending
Recursion	True

TextLabelID	DirectoryName [n]
Intended content	The name of the directory

If we take a look at the “FileNav” contentarea we see that this contentarea also contains a list. This list isn't recursive, but when you look at the figure with the screenshot above you'll see that the list contains a list of subdirectories as well as filenames. The difference is that a subdirectory triggers the subfolder link and a filename the open link. Another difference you can see in the screenshot is that a subdirectory has no size property. So you might think that we need two different lists. But if we would use two different lists, sorting them as one would be a problem. To solve this we use a single list and place all the possible elements into it.



Having all the possible element in a single list isn't such a problem since the UiaML core is meant to gather all the possible content. Which combination of content elements actually has to be shown for each list item is a question that has to be answered by one of the U*ia*ML plugins (At the time this howto was written the plugins were still in development. But answering the question at hand, should properly become possible using the logical plugin).