VISPS, a Visual System for Plan Specification

Marco Porta
Dipartimento di Informatica e Sistemistica
Università di Pavia
Via Ferrata, 1 – 27100 – Pavia – Italy
porta@vision.unipv.it

ABSTRACT
In various circumstances, it is possible to arrive at the need to specify sequences of operations that a "machine" has to perform to achieve a purpose. This paper will present VISPS, a visual system originally designed to specify mission plans for the SARA autonomous submarine robot. Although this is the particular setting for which the system had been initially devised, thanks to its flexibility it can be easily configured to adjust to different contexts and situations, even if it is always based on the same simple basic visual mechanism.

Keywords
Visual plan, planning, plan specification, task specification.

1. INTRODUCTION
In many circumstances, it is possible to run into the need to specify sequences of operations that a technological device has to perform to achieve some purpose. For example, industrial activities more and more often require computerized machineries or plants to be programmed, whose working is not determined *a priori* but can be fixed according to the particular task to be carried out. Very often, however, what such programming activities need is only the specification of a series of "elementary" operations to be completed sequentially. On the other hand, each elementary operation may be characterized by a certain number of parameters, which need to be specified for it to be correctly carried out.

Or also, in a different context, consider a robot which has to autonomously complete a "mission" composed of several objectives. Even if, as is most likely, the low-level activities determining the robot's "intelligence" do not require interventions by those who have defined the mission, there will almost certainly be the need to specify the high-level objectives characterizing its salient phases. This paper will present VISPS, a visual system to specify mission plans for the SARA autonomous submarine robot [1][3], developed as part of the Antarctic Research National Program (PNRA) promoted by ENEA (the Italian corporate body for energy, environment and new technologies). Although this is the particular setting for which the system had been originally devised, thanks to its great flexibility it can be easily configured to adjust to different contexts and situations, even if it is always based on the same simple basic mechanism.

The main idea underlying VISPS is that integration of the concepts of visual language and graphic interface can create synergies thanks to which planning the execution of a certain set of operations becomes more immediate. In a world where information technology is gaining an ever greater share of everyday life, the need to specify sequences of operations, of whatever kind, will become more and more frequent within various areas, ranging from industrial plant control to office automation. Even if the end user may not be experienced at using computers, he or she will have ever increasing freedom in defining the working modes of technological devices. VISPS has been designed just with the purpose of facilitating specification tasks, through very simple graphic formalisms, which can be easily comprehended.

2. PRELIMINARY CONSIDERATIONS
VISPS (acronym of VI*sual* S*ystem for* P*lan* S*pecification*) is conceived as a visual system for specifying plans, where by *plan* we mean any sequence of elementary operations necessary to achieve one or more objectives, of whatever kind. Here, the adjective *elementary* is used with a relative meaning, that is, it depends on the degree of conceptual abstraction we want to confer on the plan.

Generally, "high-level" elementary operations will have corresponding "low-level" *textual commands* associated with them, to be sent to the specific device that will execute the plan. These commands, which are characteristic of the elementary operations making up the plan itself, represent the "program" that the device will have to really interpret and execute. The purpose of VISPS is to facilitate choosing and organizing high-level elementary operations, by associating them with visual elements which allow their parameters to be easily specified.

In particular, as already mentioned, VISPS has been devised as a system to specify mission plans for the SARA autonomous submarine robot (where SARA stems from the Italian words S*ottomarino* A*utonomo* R*obotizzato* A*ntartico*). Therefore, a plan will be a sequence of operations necessary to carry out a mission of the robot. This, for example, might be composed of the following high-level elementary steps: (1) Select the *constant-depth* type of navigation and reach position P1; (2) Arrange yourself according to orientation O1; (3) Switch the lights on; (4) Repeat the following operation sequence for ten times: [(a) Switch the camera on; (b) Shoot the scene for fifteen seconds; (c) Switch the camera off; (d) Wait three minutes]; (5) Switch the lights off; (6) Arrange yourself according to orientation O2; (7) Switch the CTD sensor on; (8) Select the *constant-distance-from-seabed* type of navigation and reach position P2; (9) ....

The various elementary steps which compose the mission nearly always involve a certain number of parameters. For example, switching the lights on may require their intensity to be specified. Or, in a more complex case, setting up a *constant-depth* navigation type may require specifying depth, direction and speed, as well as the latitude and longitude of the arrival point.

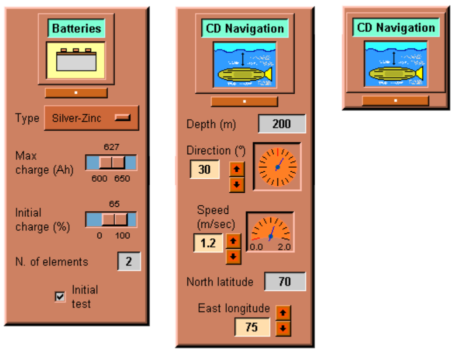
VISPS is based on the idea of defining both the various kinds of elementary operations and their parameters through particular graphic elements, which we will call *visual blocks*.

3. VISUAL BLOCKS

In VISPS, the elementary actions SARA has to perform (which may also involve on-board instrumentation) are defined through visual blocks. Substantially, a block is a rectangular-shaped graphic element, characterized by the following parameters:

- a *title*, that is, a name univocally identifying its functionality;
- a possible *icon*, which may make it more easily recognizable;
- possible *controls*, which are "instruments" that the user can exploit to supply inputs to the block (and hence, to specify parameters relative to the elementary functionality the block represents). Seven kinds of controls are provided: *Entries*, *Scales*, *Checkbuttons*, *Menubuttons*, *"Clock-like" controls*, "*Semiclock-like" controls* and *Increasable/decreasable entries*.

For example, in Figure 1-a a block is shown which has, namely (from top to bottom), the following controls: a menubutton, two scales, an entry and a checkbutton. In Figure 1-b another block is shown which has an entry, a clock-like control, a semiclock-like control, another entry and an increasable/decreasable entry.



Type Silver-Zinc
Max charge (Ah)
Initial charge (%)
N. of elements 2
✓ Initial test

Depth (m) 200
Direction (°) 30
Speed (m/sec) 1.2
North latitude 70
East longitude 75

(a) (b) (c)

Figure 1: Examples of visual blocks

Blocks having at least one control also have a particular button (the one directly below the block's icon, characterized by a small white circle in the center), which allows controls to be "hidden". Therefore, blocks can be closed at any moment, to occupy less space, and re-opened when the value of some parameter needs to be modified. For example, the block in Figure 1-b, when closed, appears as shown in Figure 1-c.

Block characteristics are defined on file, according to a very simple format. The different kinds of controls can easily be configured to adapt themselves to different types of parameters, from those requiring totally general inputs (strings) to those in which choices are imposed within predefined sets of values, both discrete and not.

3.1 Actions Associated with Blocks

Every block has certain *actions* associated with it. In principle, actions may be thought of as simple character strings. In the specific case of setting up missions for SARA, such strings will correspond to *commands* to be sent to the robot. These commands, whose syntax and semantics depend on the particular communication protocol used, are defined on file, for each block, and, as parameters, get the inputs specified by the user through the block's controls (as will be discussed in the following).

As already said, by *plan* we mean a sequence of elementary operations that SARA has to perform to complete a mission. Therefore, a visual plan for SARA will be a sequence of graphically interconnected blocks. However, since some operations might need to be executed several times, a special iterative construct is provided which makes it possible to specify cyclical behavior.

4. GENERAL DESCRIPTION OF VISPS

VISPS is totally written in Tcl [2]. The interface for specifying SARA plans appears to the user as shown in Figure 2. As can be seen, it is characterized by a *canvas* (that is, a main work area) on which the plan blocks are placed. Blocks, by means of the mouse cursor, are taken from the *choice bar* located low (which shows their titles and icons) and dragged onto the canvas, in the desired positions. Still through the mouse cursor, they can subsequently be moved directly.

The choice bar displays names and icons of the currently loaded library. However, at any time, by pressing a special button, it is possible to select new libraries to be loaded from all the available ones. Far down there is an "informative bar", which acts as a contextual help.

4.1 Using Blocks

When a block is taken from the choice bar and placed on the canvas, a *link* (a line with an arrow at one end) is automatically created between the previous block, in the sequence, and the just inserted one (unless it is the first block of a new plan which is being created). This way, the plan block chain is built. Single blocks or groups can also be inserted before or after other blocks in the plan, by means of simple *cut/copy* and *paste* operations. By pressing the right mouse button when the cursor is over a block, a contextual menu is opened containing various items, through which several actions can be undertaken.

4.1.1 The FOR Block

To allow specification of iterative behavior (that is, repeated "execution" of one or more blocks over a certain number of times), VISPS provides a special predefined block, which we will call *FOR block*. The FOR block is characterized by four *control ports*, whose purpose is to make clear which blocks pertain to the cycle and which, instead, are its input and output. Referring to Figure 2, a FOR block can be noted. In particular, block *CD Navigation* is at the input of the cycle, i.e. it is the block executed immediately before the iteration. Block *Lights ON*, instead, is at the output of the cycle, i.e. it is executed after the iteration. Lastly, blocks *Wait* and *Data Acquisition* are executed three times, as many as the value specified in the FOR block's entry.

Control ports automatically arrange themselves on the FOR block's sides, in the most "suitable" positions, depending on the positions of the blocks connected to them; this way, the plan graph turns out to be more readable.

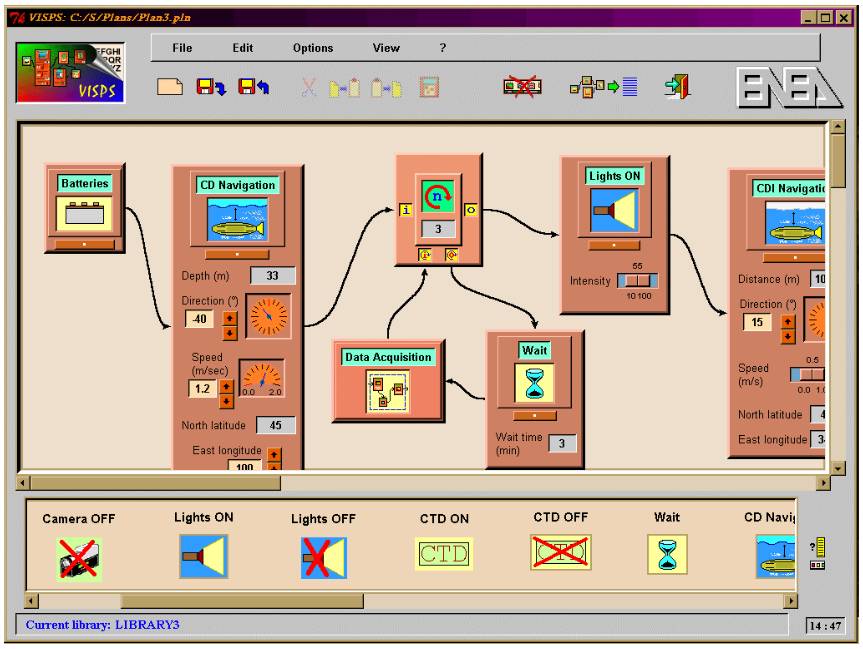


Figure 2: The VISPS interface

4.1.2 *Block Selection*

By keeping the left mouse button pressed and dragging the cursor on the main canvas, a *selection rectangle* is created: consecutive blocks within this rectangle are said to be *selected* (and graphically distinguish themselves, compared with others, thanks to a colored outline).

4.1.3 *Macro Blocks*

Since a mission plan composed of many blocks may turn out to be unclear, the VISPS system allows groups of consecutive blocks to be embodied within *macro blocks*.

Macro blocks represent an abstraction mechanism thanks to which several elementary operations can be composed to form more complex operations. To create a macro starting from a set of blocks, the set must be first selected. Afterwards, from the selection rectangle's contextual menu (or from menu *Edit* or through a special button) item *Create macro* must be chosen. Then, a dialog window will appear in which the name of the macro to be created must be entered. The previously selected blocks are thus substituted with a single block, whose title is the macro name one has just specified.

The menu of a macro block also contains item *Expand*. By selecting it, blocks in the main canvas are deleted and the sub-plan contained in the macro is displayed. "Expansion" is accompanied by a graphic effect giving the impression that the macro block is dilating. This animation effect reinforces the concept of macro block, by highlighting its meaning of "container" block. The sub-plan contained in a macro, once the expansion process has occurred, can be modified as if it were a main plan. To come back to the main plan (or to the sub-plan at the previous level, if a macro block contained in another macro block has been expanded) it is sufficient to press a special button. "Compression" too is accompanied by a graphic effect, identical but opposite to the expansion one. Recursively, macro blocks can contain other macro blocks, at any level, so that the user can enter into increasing levels of detail when building plans.

Moreover, a macro block can be inserted in any of the available libraries, so that it can be used also for building other plans. This way, the user can create, in a simple and direct manner, new functionalities, starting from the existing ones.

5. LOW LEVEL COMMANDS

Every block has a file associated with it. Its function is to define textual commands (which may depend on the block's controls) that will be generated when the visual plan is translated into the corresponding low-level textual plan. As already stated, the format of this file can be totally arbitrary and will depend on the actual device (in our case, the SARA submarine robot) which will perform the plan, and also on the particular high-level command represented by the block.

In the file, controls are referred to by means of their names, enclosed within a pair of delimiting strings (defined in the VISPS general configuration file), so that the system can unequivocally "understand" that it is managing controls. For example, we may choose to use the strings '`<<`' and '`>>`'. Thus:

```
<<North latitude>>
```

will refer, for instance, to the value of the parameter whose name is '`North latitude`' in block *CD Navigation* (Constant-Depth navigation) in Figures 1-b and 2. In addition, as it may be necessary to perform some kind of processing on parameters, files associated with blocks can also contain Tcl scripts, they too enclosed within a pair of delimiting strings (still defined in the system configuration file). For example, we may decide to use the strings '`SCRIPT{`' and '`}END`'. Then, an hypothetical low-level textual command defined as:

```
COM p SCRIPT{expr <<North latitude>> * 2}END
```

would be translated into '`COM p 90`', assuming that the value specified for parameter '`North latitude`' in the block is equal to 45 (`expr` is the Tcl command for executing arithmetic calculations). The complexity and the length of Tcl scripts can be totally arbitrary, therefore allowing, if necessary, even procedures to be defined. This means that behind a block there may also lie very complex processing activities, performed on its inputs (controls). However, the user interacts solely with the visual abstraction of these processings, that is, with the *interface* represented by the block.

Command files associated with blocks are simple textual files and, hence, can be created through any editor. However, for them to be more easily written and modified, a special editor is also provided, which is contextual to each block.

6. FROM VISUAL TO TEXTUAL PLANS

Once a visual plan has been built by composing blocks representing the various elementary steps of a mission, the corresponding low-level textual plan can be generated. This textual plan (or program) will be made up of commands defined in the files associated with each one of its blocks, and will also take into account the presence of iterations and macro blocks. After being generated, the textual plan will be sent, in some way, to the device that will have to execute it. SARA, for example, will receive commands via radio. Afterwards, typically, an interpreter will interpret and execute the low-level commands, thus producing the desired "actions".

6.1 An Example Plan

To better illustrate the process of building a visual plan and generating the corresponding textual plan, we will consider the simple example shown in Figure 3.

It is to be noted that, because of the generality of VISPS, the particular blocks presented in this example do not necessarily correspond to real functions of SARA, which, at the moment, cannot yet be precisely defined. Instead, they should be considered as examples of possible blocks to be used to build possible plans. Here we will assume that: the hypothetical block *Batteries* selects what kind of batteries to activate on the robot (supposing that there are more than one) and sets up their parameters; block *CTD ON* activates the CTD sensor to get data about physical properties of water; block *CD Navigation* sets up parameters for a constant-depth navigation and makes the robot reach a certain position; block *CTD OFF* deactivates the CTD sensor (which remains on for all the distance covered by the robot during its movements); block *Hovering* allows parameters to be set for a particular position to be reached and for a particular orientation according to which the robot must arrange itself; macro block *Filmings* allows a scene to be shot through a video camera at fixed intervals; block *FP Navigation*, lastly, sets up the coordinates of a final point the robot has to reach.

Then, suppose that the content of macro block *Filmings* is that shown in Figure 4. We will assume that: block *Lights ON* switches the lights on, according to a certain intensity; for three times, as many as specified in the FOR block's entry, the following sequence of operations is executed: the camera is turned on (block *Camera ON*), the robot waits one minute (block *Wait*), the camera is turned off (block *Camera OFF*) and the robot waits five minutes (block *Wait*); at last, once the iteration is over, block *Lights OFF* switches the lights off.

Now, again for illustrative purposes, suppose that low-level command files associated with each block contain simply the name of the block with which they are associated, followed by the sequence of the values specified for its parameters. This means, for example, that for block *FP Navigation* the structure of the file is:

```
FP Navigation: <<North latitude>>,
               <<East longitude>>, <<Depth (m)>>
```

Then, the textual plan generated for the visual plan in Figure 3 will be the following:

```
  Batteries: Silver-Zinc, 629, 50, 1, 1
  CTD ON: 5
  CD Navigation: 80, -30, 1.3, 71, 165
  CTD OFF
  Hovering: 71, 153, 80, 15, 10, 0
  Lights ON: 83
  Camera ON
  Wait: 1
  Camera OFF
  Wait: 5
  Camera ON
  Wait: 1
  Camera OFF
```

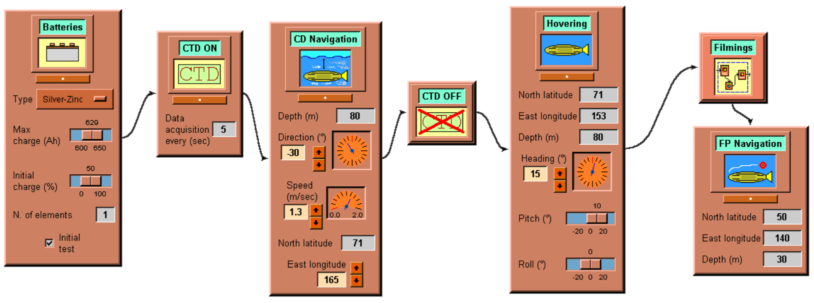


Figure 3: An example plan

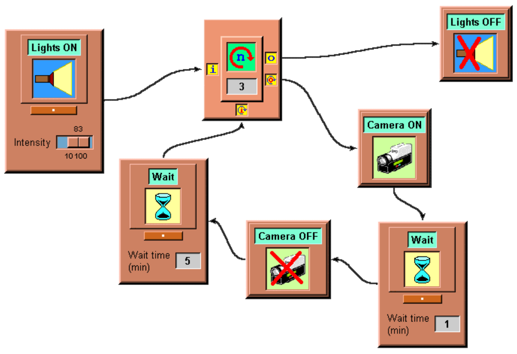


Figure 4: Content of macro block *Filmings* of Figure 3

```
Wait: 5
Camera ON
Wait: 1
Camera OFF
Wait: 5
Lights OFF
FP Navigation: 50, 140, 30
```

7. CONCLUSIONS

We think that the choice to associate a graphic control with every parameter, which can be directly manipulated within the block, is very interesting. In fact, the visual block in itself represents an *interface within the interface*, that is, a communication means between the user and the single step of the mission. We hold that this aspect, together with the possibility to create macro blocks from elementary blocks, allowing new functionalities to be easily added to the existing ones, can greatly facilitate the task of those who have to specify mission plans, independently of their particular type.

8. REFERENCES

[1] Bono, R., Caccia, M., and Veruggio, G., Simulation and Control of an Unmanned Underwater Vehicle. IEEE International Conference on Robotics and Automation, Nagoya, Aichi, Japan, 21-27 May 1995.

[2] Ousterhout, J. K., Tcl and the Tk Toolkit. Addison Wesley, 1994.

[3] Papalia, B., Work in progress on the antarctic underwater robot SARA; problems of navigation below the ice cover. International Workshop Ross Sea Ecology, Taormina, Italy, 14-16 May 1996.