User Guide of the
Workload Analyser Tool within IMSE

The Integrated Modelling Support Environment project

Lionel Mallet
Luisa Massari
Paola Rossaro

R4.2 – 5 Version 1

August 1991

Distribution level All IMSE project teams and CEC

Approved by

The IMSE Project The IMSE project is a collaborative research project supported by the CEC as ESPRIT project no. 2143. It is being carried out by the following organisations :- BNR Europe STL, Thomson CSF, Simulog A.S., University of Edinburgh, INRIA, IPK (Berlin), University of Dortmund, University of Pavia, SINTEF (University of Trondheim), University of Turin and University of Milan.

Contents

References

[1] Maria Calzarossa and Luisa Massari, Requirement Specifications of the Workload Analyser Tool. Document IMSE R4.2 – 1, University of Pavia, 1989

[2] Maria Calzarossa, Luisa Massari and Giuseppe Serazzi, The Workload Analyser Tool - User Interface. Document IMSE R4.2 – 2, University of Pavia, 1990

[3] Maria Calzarossa, Luisa Massari and Giuseppe Serazzi, The Design of the Workload Analyser Tool. Document IMSE D4.2 – 1, University of Pavia, 1990

[4] Sylvan Dissoubray, The Queue Network Tool Design Document. Document IMSE R5.2 – 4, Simulog S.A., 1990

[5] GIE Emeraude, Emeraude manual, Syntagma System Literature, Truro, UK, 1990

[6] Jane Hillston and Neil Stevenson, The Design of the Experimenter. Document IMSE R3.1 – 4, University Edinburgh, 1990

[7] Andreas L. Opdahl and Vidar Vetdland, Evaluation of EXT and WAT design. Document IMSE R6.6 – 6, SINTEF, 1990

[8] Graham Titterington and Andy Thomas, The IMSE Object Management System. Document IMSE R2.1 – 5, STC Technology, 1990

[9] Maria Calzarossa, Lionel Mallet, Luisa Massari and Giuseppe Serazzi, The Workload Analyser Tool - User Guide. Document IMSE R4.2 – 4, University of Pavia, 1991

[10] Andy Thomas, The Graphical Workbench. Document IMSE R2.1 – 8, BNR Europe, 1991

[11] Chris Uppal, Design of the IMSE SDMF. Document IMSE R2.2 – 3, STC Technology, 1991

1 The Integrated Modelling Support Environment

The design and modelling of computer systems often require the use of several tools, with different user interfaces and support environments, which are sometimes not easily understandable and usable.

The Integrated Modelling Support Environment (IMSE) allows the user to perform all the necessary tasks for system designing, modelling and evaluation in an homogeneous environment, by integrating several tools in a common workbench. IMSE is an object oriented environment with a graphical interface. It runs on top of the Emeraude environment [5], an implementation of the PCTE Public Interface Specification (PCTE is published under the responsibility of the PCTE Interface Management Board).

Applications, data and storage of files are seen as objects identified by their type and their name.

The tools of IMSE allow the user to perform various activities related to the modelling process, such as system description, model construction, experimentation and analysis.

There are six main modules, as shown in Fig. 1.

The `Workbench` provides the common interface for the tools in IMSE: the user can create, select and eventually modify objects; furthermore, it is possible to call, control and display models execution.

The `Object Store` is the database of the environment. All the objects in IMSE are stored in this module, which provides facilities and routines for their management.

The `System Description` supports the specification of software and hardware modules and the construction of workload models.

The `Performance Modelling` allows system models construction using different solution paradigms.

The `Experimentation` provides support for designing and executing experiments with models, and for data interchange.

The `Reporter` is aimed towards producing documents, and provides all the facilities for statistical data processing in order to have a better view of the data.

Each tool is represented as an object with its own rules and operations. Therefore, the user has to create and edit the object in order to give the necessary instructions and information about input data and execution requirements.

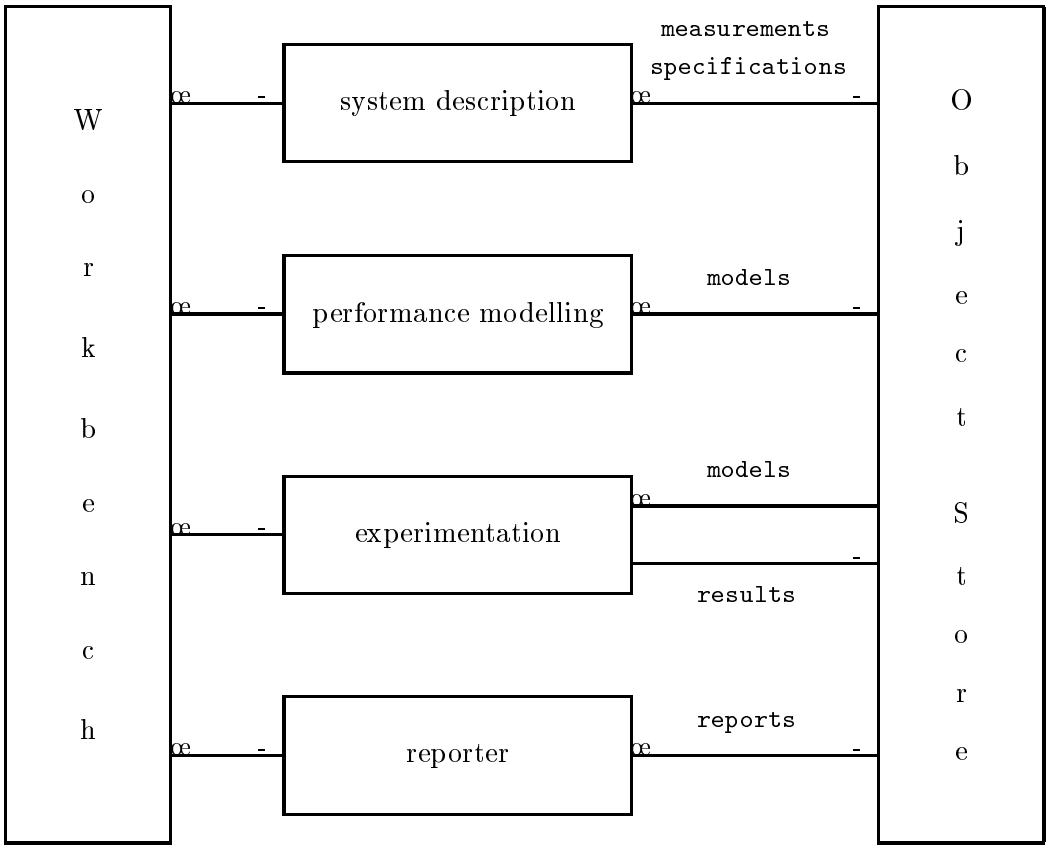


Figure 1: IMSE architecture.

The model execution is performed by the Experimenter [6], which can be used with different purposes, such as evaluation of a design, validation of a model and debugging.

Note that all the system runs on Sun computers and is built on top of the SDMF [11].

2 The Workload Analyser Tool

Workload characterization is a fundamental part of performance evaluation. The Workload Analyser Tool (WAT) [3] available within IMSE can be used to derive a manageable and more compact representation of the workload, in that it allows the construction of workload models whose characterizing parameters can be used by other tools in the system.

WAT processes raw input data collected by means of the measurement tools and the routines available on the target systems, and identifies, by means of suitable statistical techniques, a set of groups (classes) of components having homogeneous characteristics.

Figure 2 shows the overall structure of the tool. Measurements represent the input from external world to IMSE. WAT outputs consist of parameters, characterizing workload models, whose values can be the input for the modelling tools of IMSE.

The Workload Analyser Tool has been designed to be independent of the type of data streams to be analyzed; the input of WAT can consist of accounting files measured on systems running under various operating systems (i.e., UNIX, VME) as well as ASCII or binary files.

The user has to specify the structure of the records in the ASCII/BINARY input files, i.e., the format of each variable (parameter), and has also to select the variables to be used for the description of each workload component.

It is possible to specify all these information (e.g., mnemonic names of the variables, type, length) the first time a certain type of input data is selected and store them in a library of record structures, i.e., formats.

Another relevant feature included in WAT is the possibility of directly processing accounting files measured on systems running under UNIX or VME operating systems. A lot of preliminary work is required to convert such files in a "readable" format since, for efficiency reasons, they are usually stored in a "compressed" format not easily "decodable".

For saving disk space, the WAT can accept input from either disk or tape.

After the input description and the selection of the characterizing parameters of each workload component, the "real" processing of workload data begins. The basic statistical descriptors, such as, mean, variance, standard deviation, skewness, kurtosis, maximum and minimum values, are provided.

A preliminary analysis of the input data, which includes application of various types of transformation, trimming of the outliers and sampling, is performed. All these operations are mainly aimed towards the clustering analysis, which represents the core of the WAT since it allows the construction of workload models and the identification of the input parameters for the corresponding system models, designed by other tools available within IMSE.

As seen within IMSE, WAT is divided in two logical parts, that correspond to input definition and data processing.

The first phase is performed by the `edit` operation on the `Wat_model` object, the second one by the Experimenter, a common environment that allows the execution of the different tools and the communication among them through some data interchange facilities.

In the following sections we will describe the steps needed for the workload analysis, that is how the user can define input parameters, and run the execution.

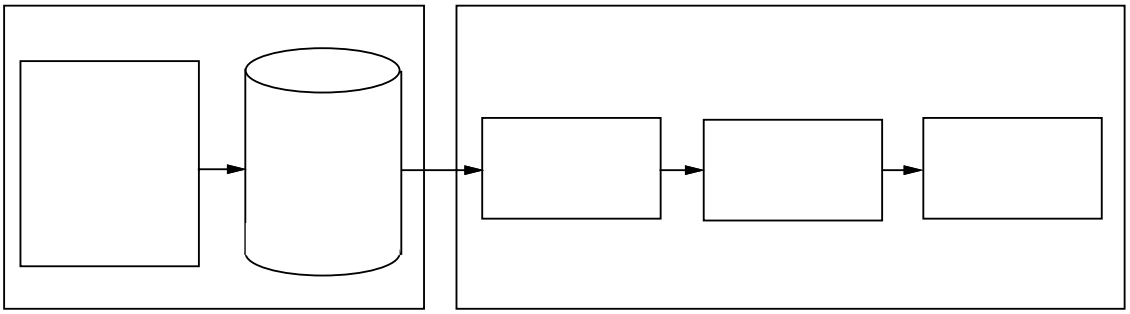


Figure 2: Rôle of the Workload Analyser Tool with respect to IMSE and the external world.

3 Creating a WAT model

The user of WAT has to provide the specifications of the analysis, i.e., input and output specifications, and the description of measurement data by means of the WAT Editor. Therefore, it is necessary to have a `Wat_model` type object in the IMSE workbench [10].

For a better understanding and use of the components of the graphical interface of IMSE tools, refer to Appendix A.

By clicking the right mouse button with the mouse cursor located on the workbench canvas, a menu containing the `Create object` item will be displayed. As a consequence, a window labelled "Create Object" (see Fig. 3) is opened.

Figure 3: Creating an object in IMSE workbench.

The user can choose the `Object type` (sum item). In our case we have `Wat_model`. The `Object name` (string item) and `Object version` (integer item) identify a unique object of the same type. `Directory` is a string item specifying the IMSE pathname where the object has to be stored. The `Abort` button discard all the operations. To save the data and to create the object, the user has to select the `Done` button. The new object will then appear as an icon on the workbench canvas, under the specified directory, as shown in Fig. 4.

4 Editing a WAT model

Once the object is created, the user has to select the `edit` item within the object menu (see Fig. 5). It calls the graphical interface - WAT Editor - which allows the user to define the various operations to be performed on the input data.

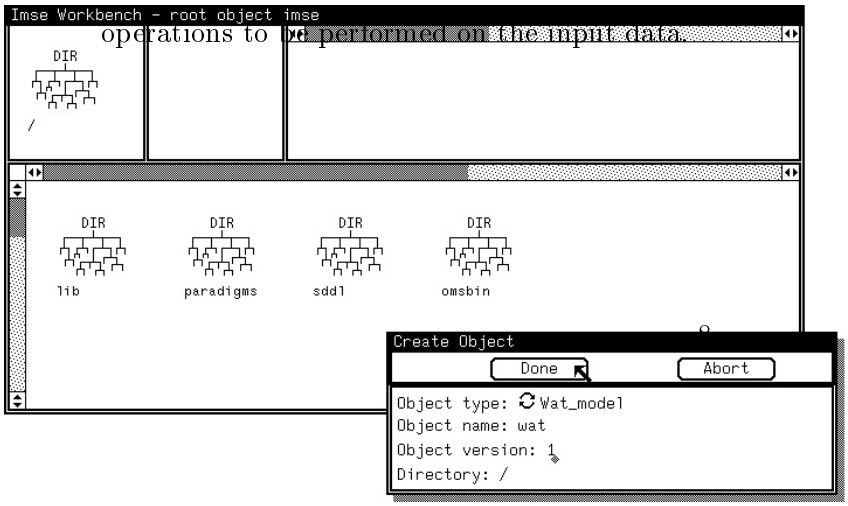


Figure 4: `Wat_model` object in IMSE workbench.

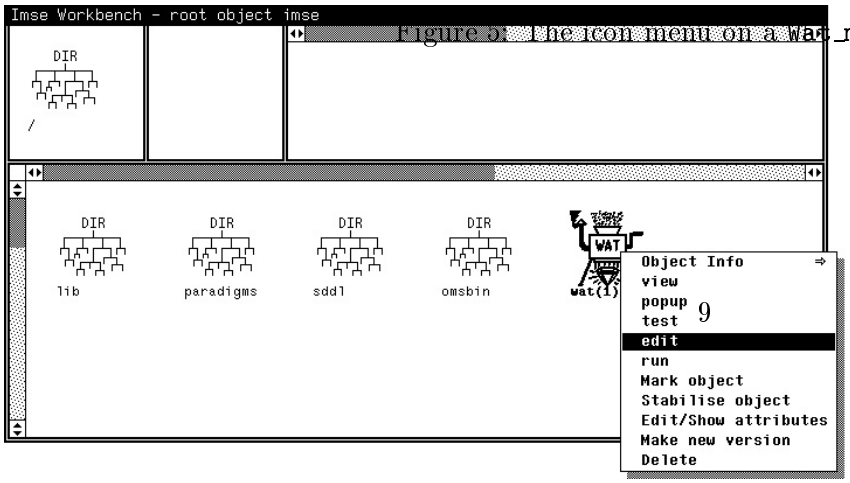


Figure 5: The icon menu on a `Wat_model` object.

4.1 The "WAT Editor window"

The interface starts by displaying a window labelled "WAT Editor window"(see Fig. 6).

This window, which gathers a few preliminary information, is logically subdivided in four sections related to `Input Specifications`, `Control Specifications`, `Display Specifications` and `Output Specifications`, respectively. Each section has its own help on line which is available through the `Help` button.

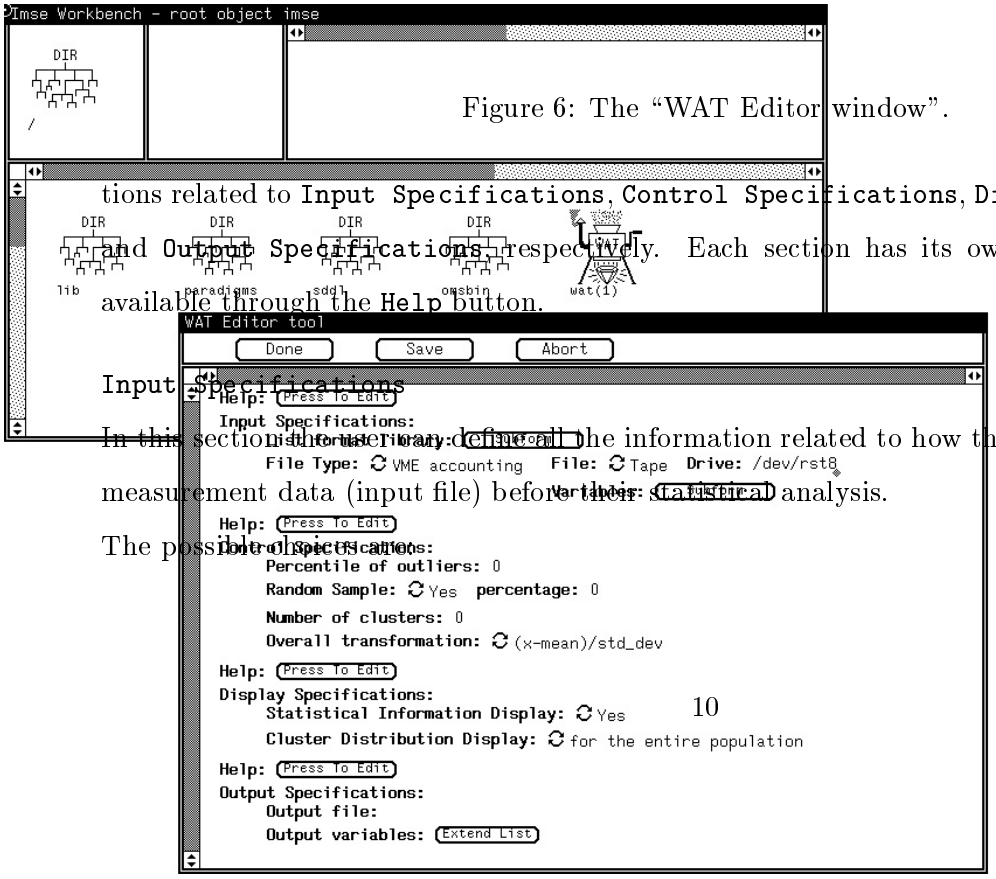


Figure 6: The "WAT Editor window".

`Input Specifications`

In this section the user can define all the information related to how the tool has to manage the measurement data (input file) before their statistical analysis.

The possible objects are:

- `List of format library`: the user can list the contents of the library of ASCII/BINARY formats. A specific window is opened (see Sect. 4.1.1);

- `File type`: it allows the user to specify the format of the input data (i.e., the organization of the data in the input file); this sum item gathers the following standard formats: UNIX accounting, VME accounting, user pre-defined formats (found in the proper format library and displayed by name) and new `ASCII/BINARY` formats.
  Let us examine in details the various options:

  1. **accounting format:** the user has the following possibilities within the sum item `File`:
     - `disk`: the file is stored on disk; the name of the file containing the data (string item `Filename`) must be specified;
     - `tape`: the file is stored on a tape; the user has to specify the name of the device driver associated to the tape (string item `Drive`);
  2. **user pre-defined format:** the name of the file containing the data (string item `Filename`) is required;
  3. **new `ASCII/BINARY` format:** the user has to specify the name of the file containing the measurement data (string item `Filename`). The structure of the file, i.e., the format, is defined in the "Format" window (see Sect. 4.1.2).

In all the three cases, the user will also have to specify the information of the parameters he wants to analyse (see Sect. 4.1.3).

`Control Specifications`

This section is concerned with the information related to the analysis of the measurement data, namely:

- `Percentile of outliers`: the percentile of the distribution of each variable for trimming of the outliers is specified (integer item); it is also possible to set it as an undefined value,

that is, as a `FREE variable` (see Sect. A.7), and to specify it at a later stage, e.g., during the experiment specification (see Sect. 6);

- `Random Sample`: the user can select a random sample (sum item); an integer item is displayed to choose the percentage of data of the sample; it is also possible to set the percentage as an undefined value, that is, as a `FREE variable` (see Sect. A.7), and to specify it at a later stage, e.g., during the experiment specification (see Sect. 6);
- `Number of clusters`: an integer item allows the user to specify the maximum number of clusters he wants to obtain with the cluster analysis (see [3] for more details); it is also possible to set it as an undefined value, that is, as a `FREE variable` (see Sect. A.7), and to specify it at a later stage, e.g., during the experiment specification (see Sect. 6);
- `Overall transformation`: a sum item for the selection of the transformation to be applied to all the variables in order to scale them to the same range before the cluster analysis.

`Display Specifications`

This section deals with:

- `Display of Statistical Information`: the user can select to display the statistical information during the preliminary analysis of the data (sum item);
- `Display of Cluster Distributions`: the distributions of the variables can be displayed for the entire population or for each cluster produced.

`Output Specifications`

This sections deals with:

- `Output object`: not yet implemented;
- `Output specifications`: a list item of the possible output parameters (i.e., center, minimum, standard deviation, and so on) for each cluster in the optimal partition(s); by default all the possible values are stored.

4.1.1 "List of format library" window

Figure 7: The "List of format library" window.

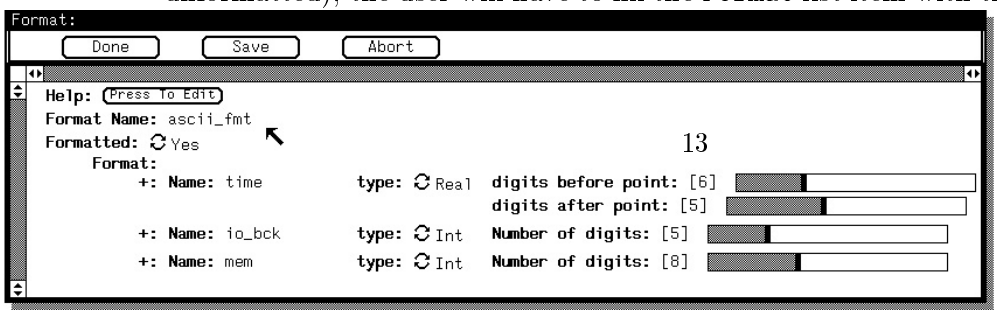
The user can list the contents of the format library, one format at a time. The name of the format he wants to display has to be selected through the first sum item.

4.1.2 "Format" window

Figure 8: The "Format" window.

This window contains its own help, available, as usual, through the `Help` button located at the top of the window, and a few items to define the specific format, namely:

- `Format Name`: name of the format (string item);
- `Formatted`: a sum item for the specification of the type of record structure, i.e., formatted (ASCII), or unformatted (binary);
- `Format`: list of the format of each variable. Depending on the previous choice (formatted or unformatted), the user will have to fill the `Format` list item with the name of each variable



(string item), its type (sum item), i.e., real or integer, and, in the case of formatted records, with the length (i.e., number of digits) of the field associated to the variable itself (integer item).

All the specifications are saved by selecting either the `Done` or `Save` buttons. Alternatively, they can be discarded by selecting the `Abort` button.

WARNING

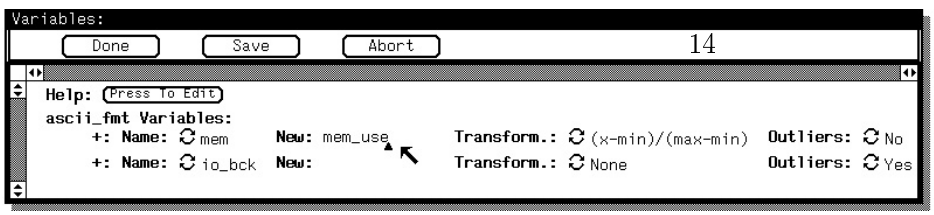
The save operation does not write permanently the format information in the library. This operation simply means "keep the information for the current session". The format will be saved in the library when the user will save the whole specifications in the "WAT Editor window".

4.1.3 "Variables" window

Figure 9: The "Variables" window.

This window contains its own help, available, as usual, through the `Help` button, and a list item to specify the parameters the user wants to analyse and how to perform this analysis.
This list consists of four components:

- `Name`: name of the variable; this item is a string item in case of a new ASCII/BINARY format or a sum item otherwise. Let us remark that for a new ASCII/BINARY format the name must correspond to a name of the variable given in the format definition;
- `New`: to associate a new name to the variable (string item);
- `Transform.`: to specify the transformation(s) to be applied to each variable (sum item);



- `Outliers`: to specify if the user wants to remove the outliers for a specific variable (sum item).

The `Done` or `Save` buttons and the `Abort` button allow the user to save and discard all the selections, respectively.

5 Running a WAT model

IMSE supplies a common environment (the Experimenter) which acts as an interface for all the applications that need to run an experiment, separating the execution and the model definition phases. With the same generic operation it is possible, for example, to execute different tools, to exchange information among them, to define compound and conditional experiments.

In order to run a WAT model, the user has to create a `Plan` object, where the frame for the experiment will be defined, following the same steps as for the `Wat_model` object creation (see Sect. 3). The creation of a `Plan` object implies also the `edit` operation, which opens a canvas labelled "Experimental Plan Editor".

5.1 Editing a `Plan` object

In the Experimental Plan Editor canvas the user has to create a `MODEL` node, by using the middle mouse button with the mouse cursor on the empty canvas and by selecting the icon labelled "`MODEL`" (see Fig. 10).

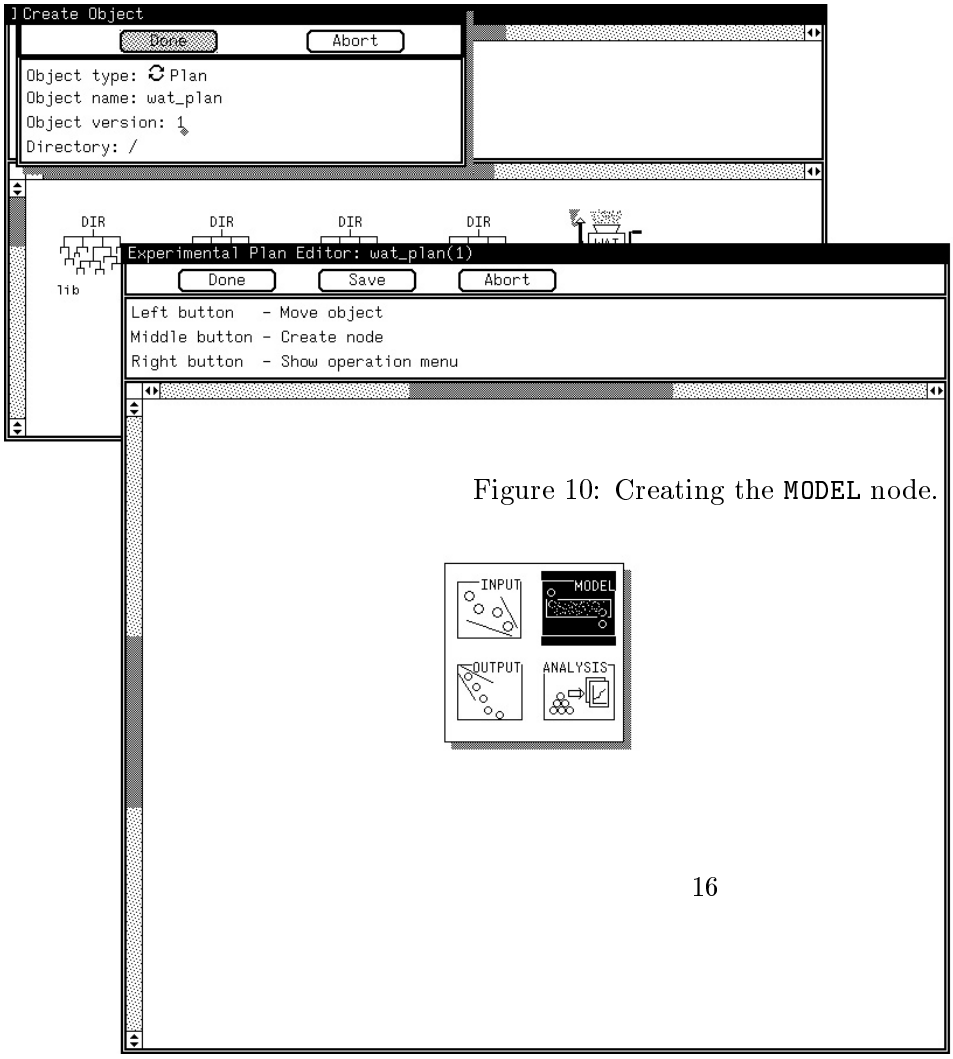


Figure 10: Creating the `MODEL` node.

5.1.1 Setting the attributes

Once the `MODEL` node icon is selected from its menu, the user can choose the `Set node attributes` item and open an "Attributes" window (see Fig. 11) which contains some information about the model to be run.
More precisely, we have the following information:

- `Local Name`: a string item that identifies the current frame;
- `OMS Name`: a string item for the name of the object to be run (in our case the newly created `Wat_model`), including the version number.

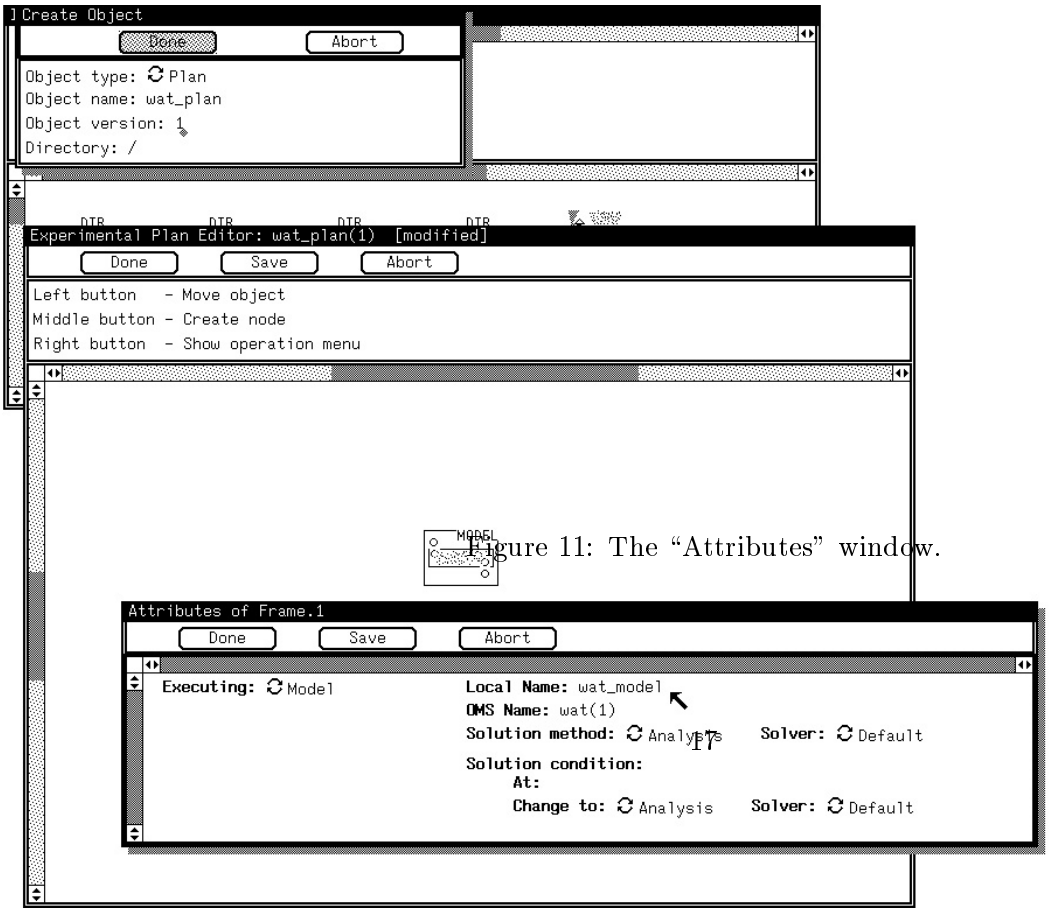


Figure 11: The "Attributes" window.

For a simple run of the experiment the user can leave all the other values of this window unchanged. The details about more complex executions are out of the scope of this report (refer to Experimenter documents for more details).

5.1.2 Loading the specifications

The selection of the `Load specifications` item on the `MODEL` node menu will attach two other nodes to it, the `INPUT` and `OUTPUT` nodes, respectively.

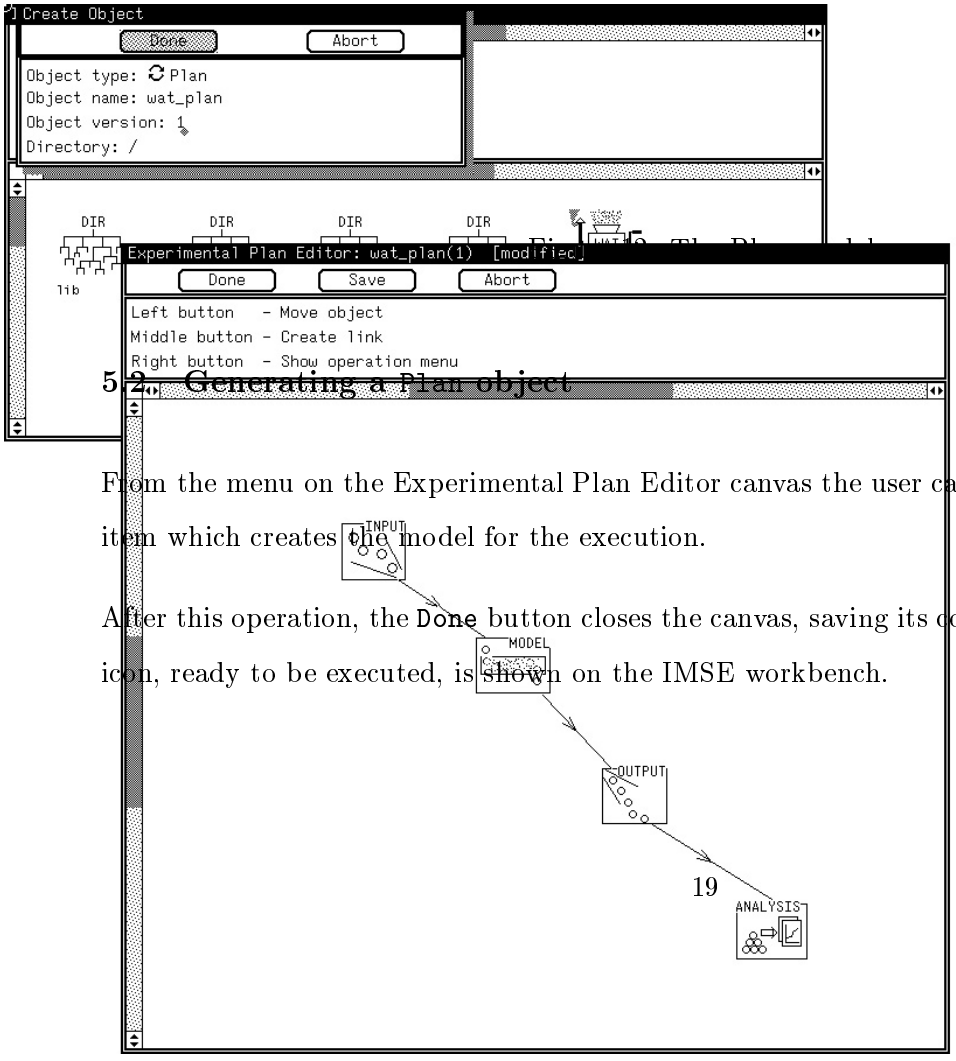
5.1.3 The `INPUT` node

The `INPUT` node contains the values left undefined in the WAT editing session, such as `Number of clusters`, which the user can see by using the `Set node attributes` item on the menu of this node.
The "Attributes" window contains a list of the undefined values identified by their label (`Parameter Name`) and their type (`Parameter Value`). A default value is associated to this list. However, it can be modified by the user by setting one or more values from a list item menu (see Sect. 6).

5.1.4 The `OUTPUT` node

It contains the list of the output variables selected in the WAT editing session. By using the `Set node attributes` item, the user can display the list and, by switching the `Selected` sum item, he can choose which values are to be displayed as outputs.

In order to complete the model definition the user needs to add another node, labelled `ANALYSIS`, to be linked to the `OUTPUT` node, as shown in Fig. 12. The `Done` and `Save` buttons in each window save the selections, while the `Abort` button discards them.



5.2. Generating a `Plan` object

From the menu on the Experimental Plan Editor canvas the user can select the `Generate Plan` item which creates the model for the execution.

After this operation, the `Done` button closes the canvas, saving its contents, and the `Plan` object icon, ready to be executed, is shown on the IMSE workbench.

5.3 The Run operation

The user has to select the `run` item from the menu of the `Plan` object in the IMSE workbench (see Fig. 19). After a few preliminary operations related to the support environment (see [6]), the WAT execution starts, following the specifications selected in the WAT and Plan editing sessions. During the execution a few objects are created which are not visible from the IMSE directories. They can be displayed by following the object links.
In the `Wat_model` object a link to a `Run` object is created which in turn is linked to an `Output` object, containing the WAT results. The user can display these values by selecting the `view` item on the `Output` object icon menu. The outputs can be used as input for other tools, as we will explain in the example (see Sect. 6).

6 An example

In order to make clearer the description of the integrated version of the WAT within IMSE and to make easier its use, we will present an analysis performed on a real set of data.
The problem we approach in this example is the analysis of measurements collected on an Ethernet local area network. The workload model obtained by the WAT provides the characterizing values used by QNET [4] as input parameters of the queueing network model of the Ethernet.
The data coming from the external world refer to the packets sent over the network from either terminal servers and disk servers. Among the various information describing each packet we have selected two parameters:

- packet length (expressed in bytes),
- interarrival time (expressed in seconds).

Figure 13 shows the structure of the file containing these data which will be taken as input by the Workload Analyser Tool. The first column represents the packet length, named `length` in what follows (integer of four digits, see format description in Fig. 15), and the second one is the interarrival time, named `delta` (real of two and four digits before and after the point,

respectively, see format description in Fig. 15).

In what follows, we will present a few figures displaying the sequence of the operations to be performed in order to specify the parameters of the analysis.

```
  74 0.0009
  60 0.0031
1345 0.0034
  60 0.0033
  60 0.0017
1345 0.0049
  60 0.0005
  60 0.0008
  64 0.0021
 577 0.0006
  60 0.0027
1345 0.0068
  64 0.0006
  60 0.0010
  60 0.0010
 104 0.0049
1345 0.0018
```

Figure 13: Example of the input file for the WAT.

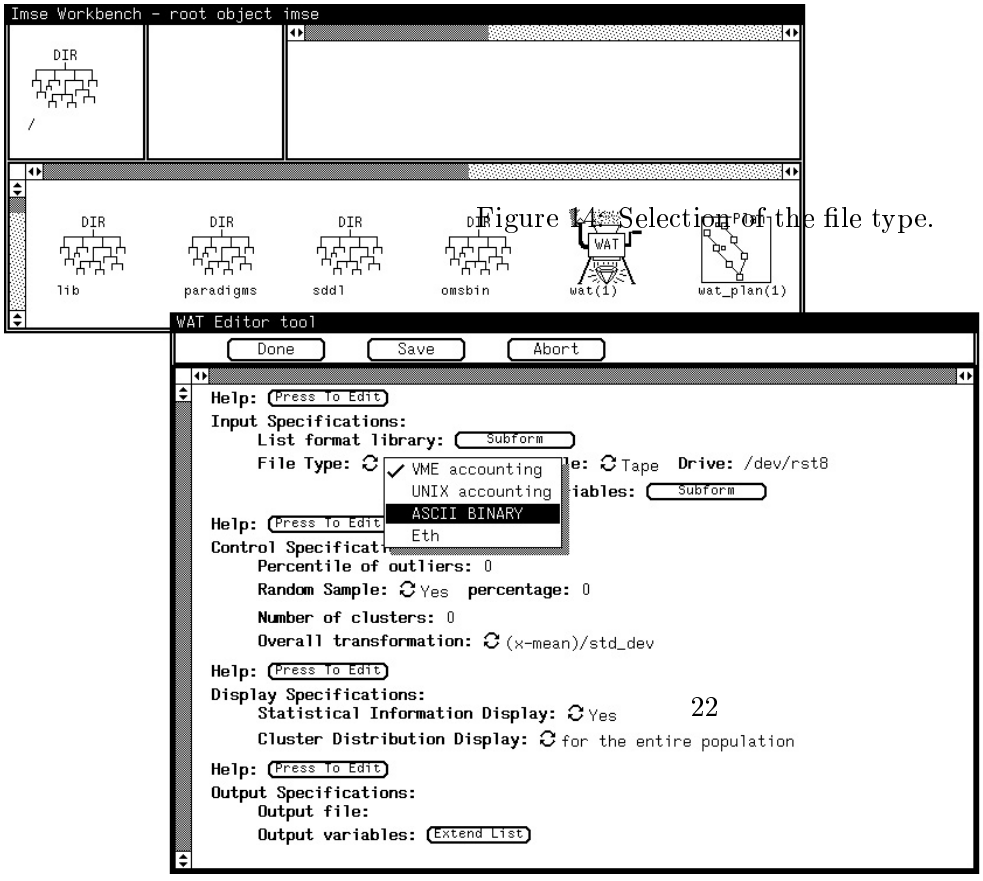


Figure 14: Selection of the file type.

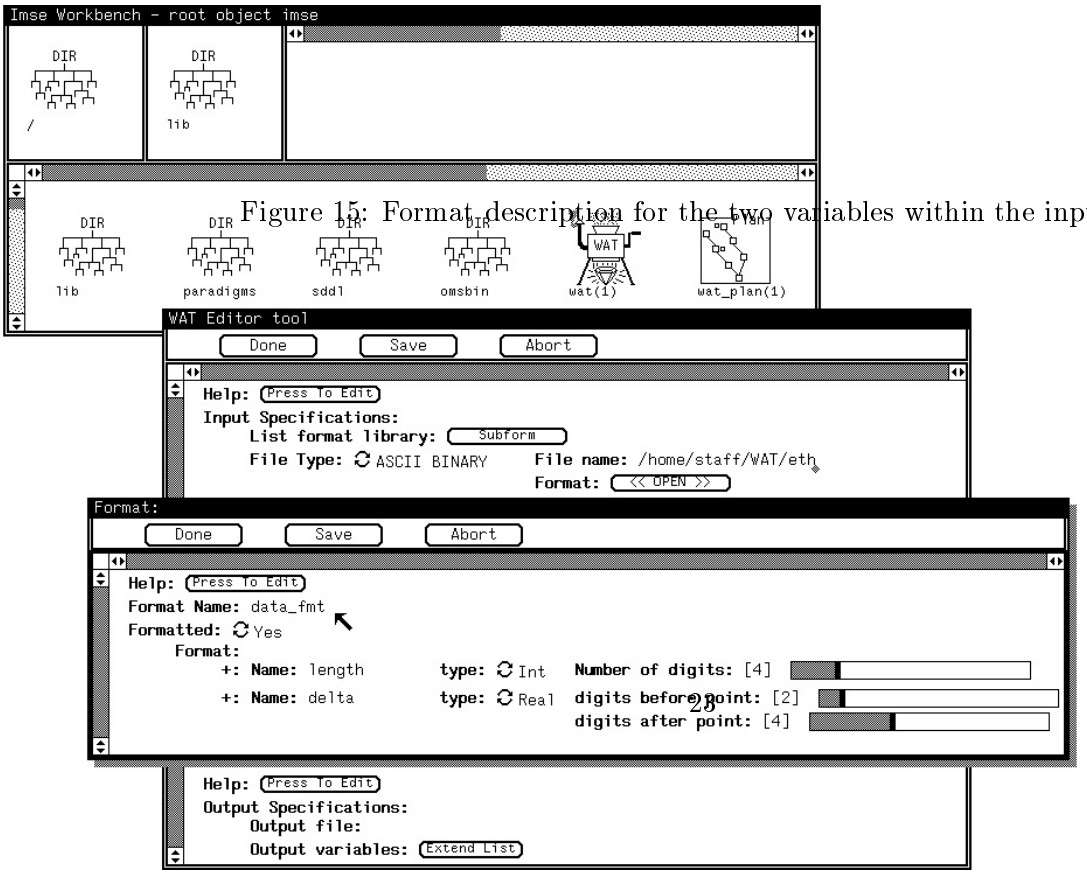


Figure 15: Format description for the two variables within the input file.

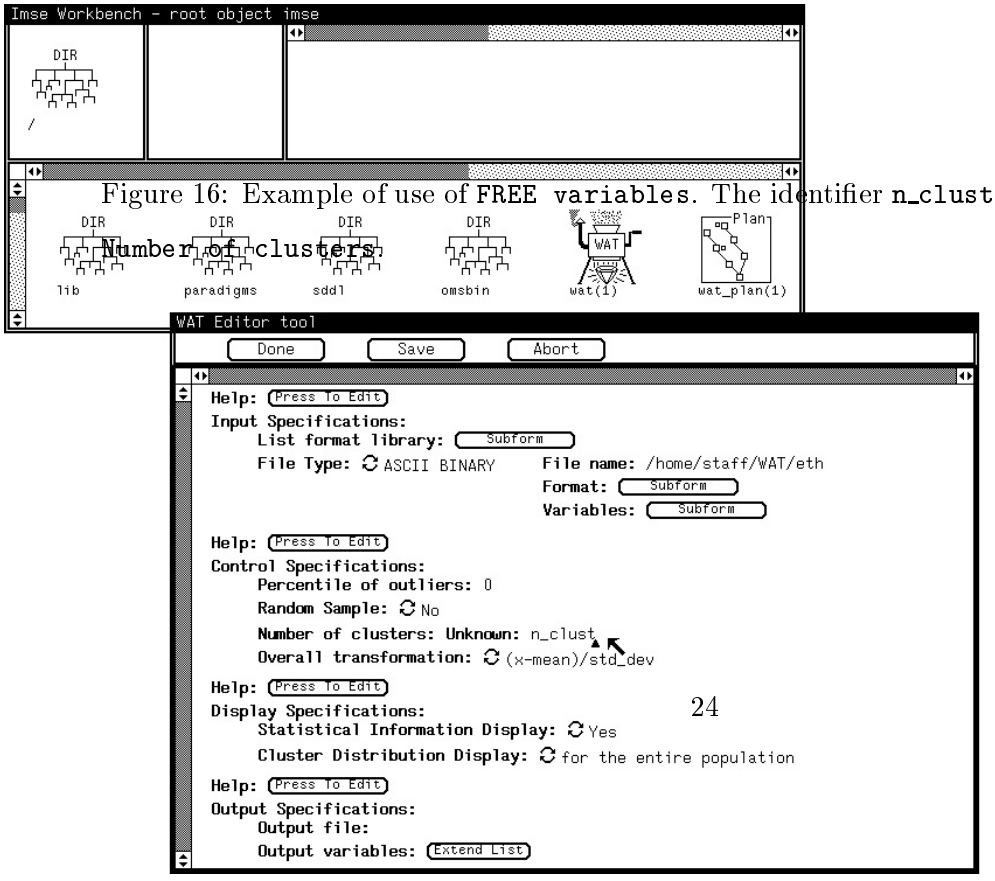


Figure 16: Example of use of `FREE variables`. The identifier `n_clust` is associated to the item `Number of clusters`.

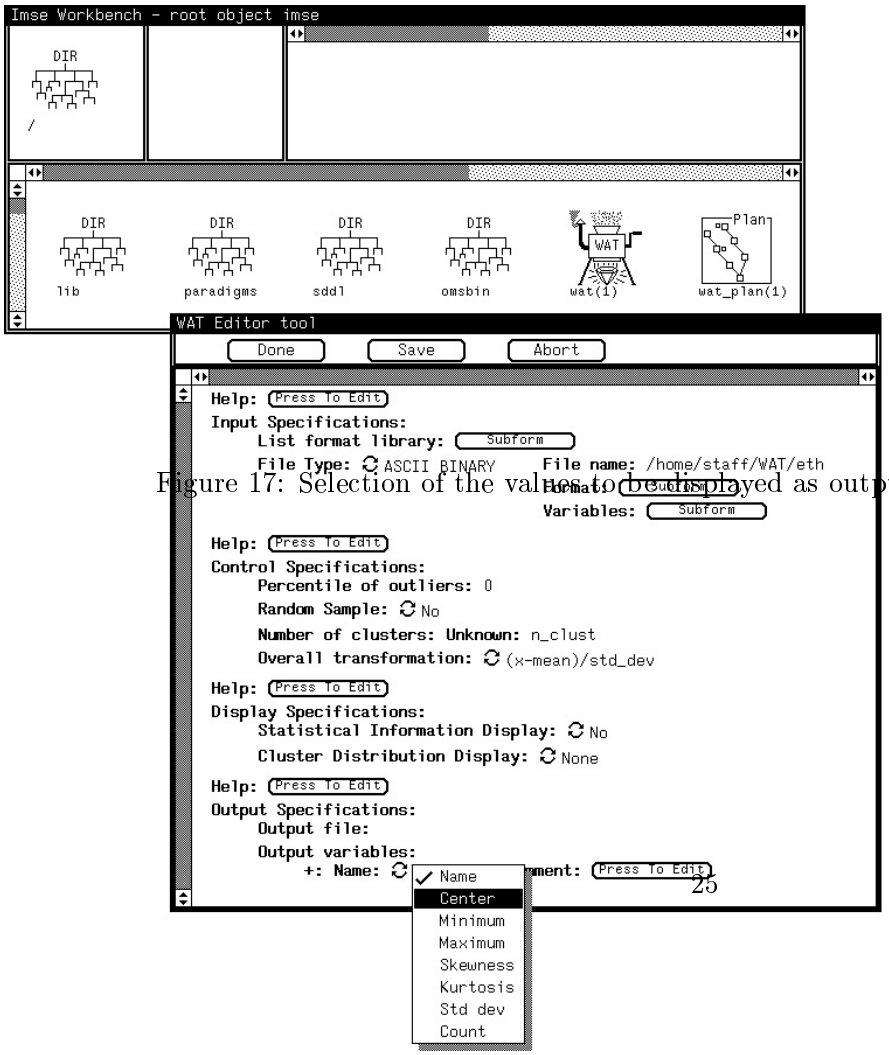


Figure 17: Selection of the values to be displayed as outputs (in our case the `Center`).

Once the specifications have been defined, the user can create and edit a `Plan` object. Figure 18 shows how to set the value of the `Number of clusters`, left undefined in the WAT editing session.

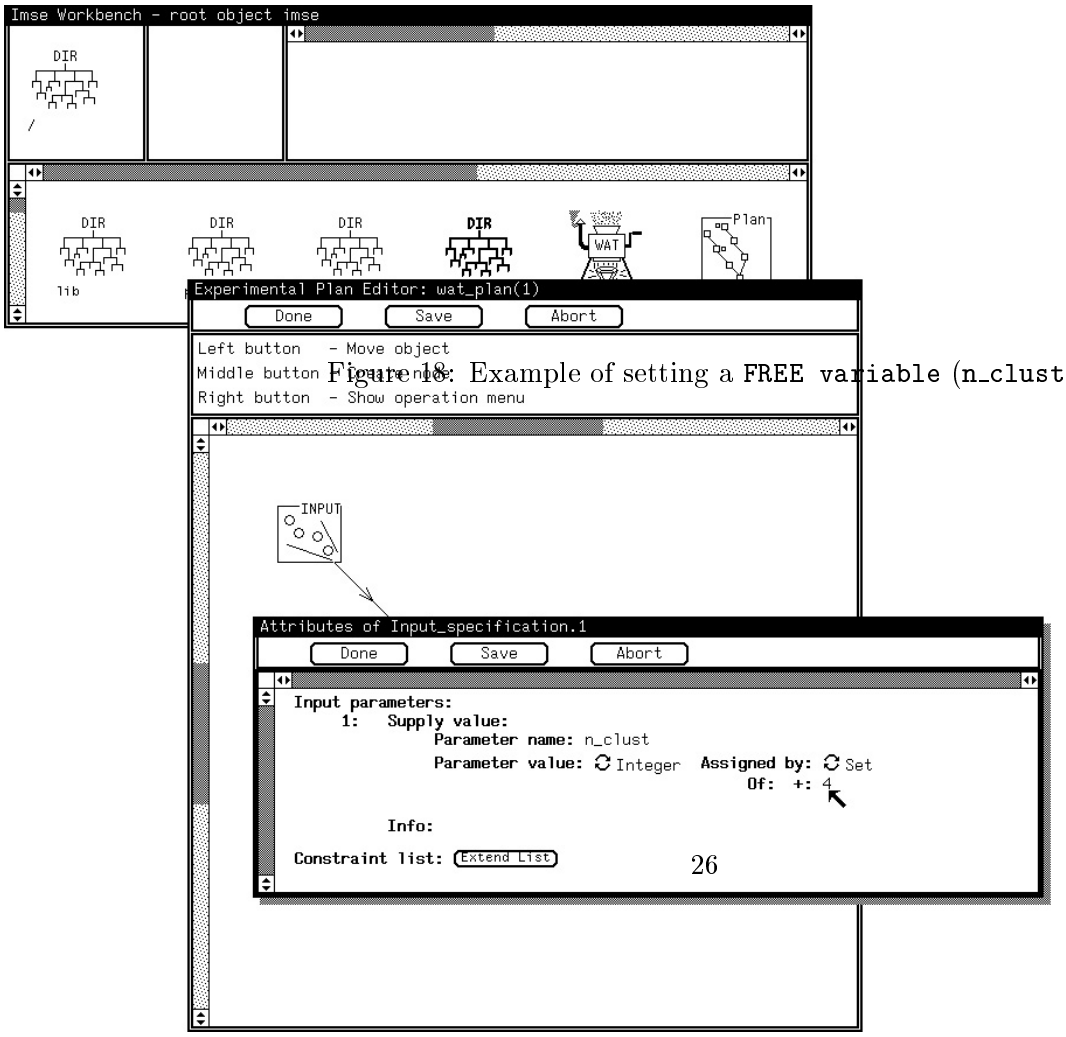


Figure 18: Example of setting a `FREE variable` (`n_clust`).

After exiting from the Experimental Plan Editor, the `run` item selection will cause the WAT execution to start (see Fig. 19).

Figure 19: Run of the WAT.

The results obtained from the WAT run, using the data shown in Fig. 13, can be seen in the `Output` object, reachable following the links from the `Wat_model` object through the `Run` (see Figs. 20 and 21).

Figure 20: The link from the `Wat_model` to the `Run` object.

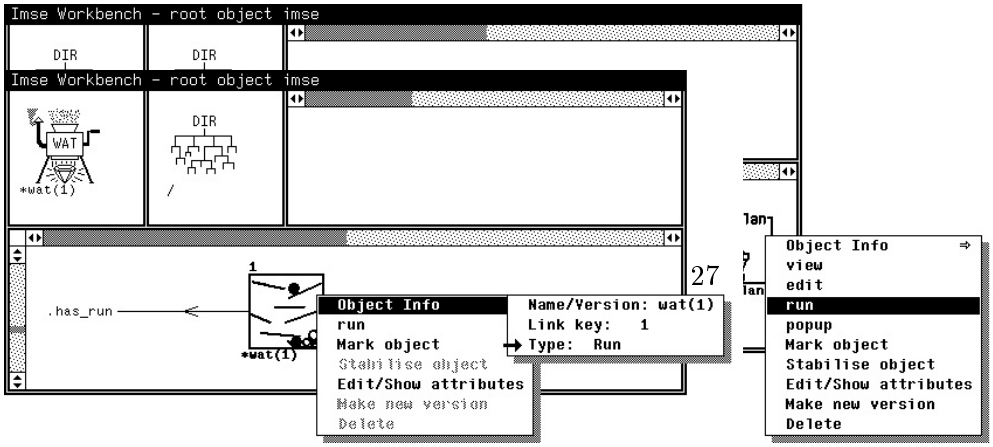
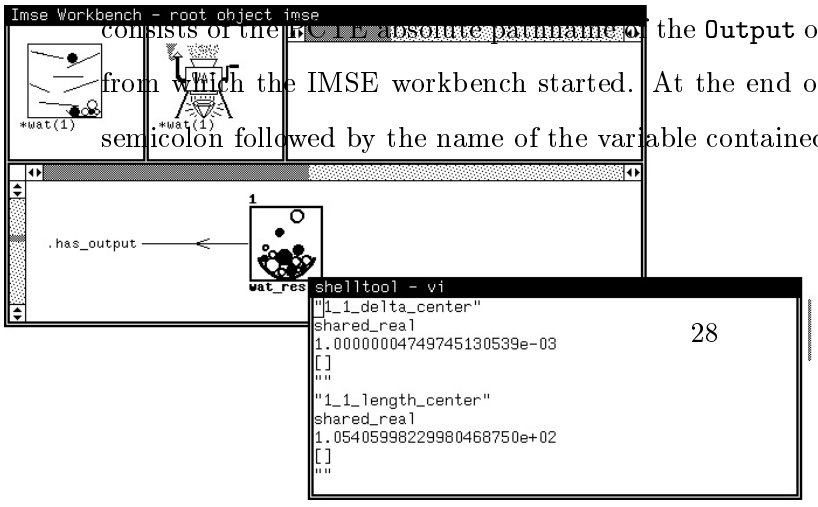


Figure 21: The link from the `Run` to the `Output` object.

The values within the `Output` object (see Fig. 22) can be used now as input for modelling tools of IMSE, such as the QNET, using the dot notation facility.

Figure 22: The output values, written in SDDL code, within the `Output` object.

Figure 23 shows the Plan Editor with a QNET model. In the `INPUT` node the values are changed with a string item referring to the `Output` object generated by WAT. The first part of the string consists of the UNIX absolute pathname of the `Output` object, which can be read in the window from which the IMSE workbench started. At the end of the pathname, the user has to add a semicolon followed by the name of the variable contained in the `Output` object (see Fig. 23).



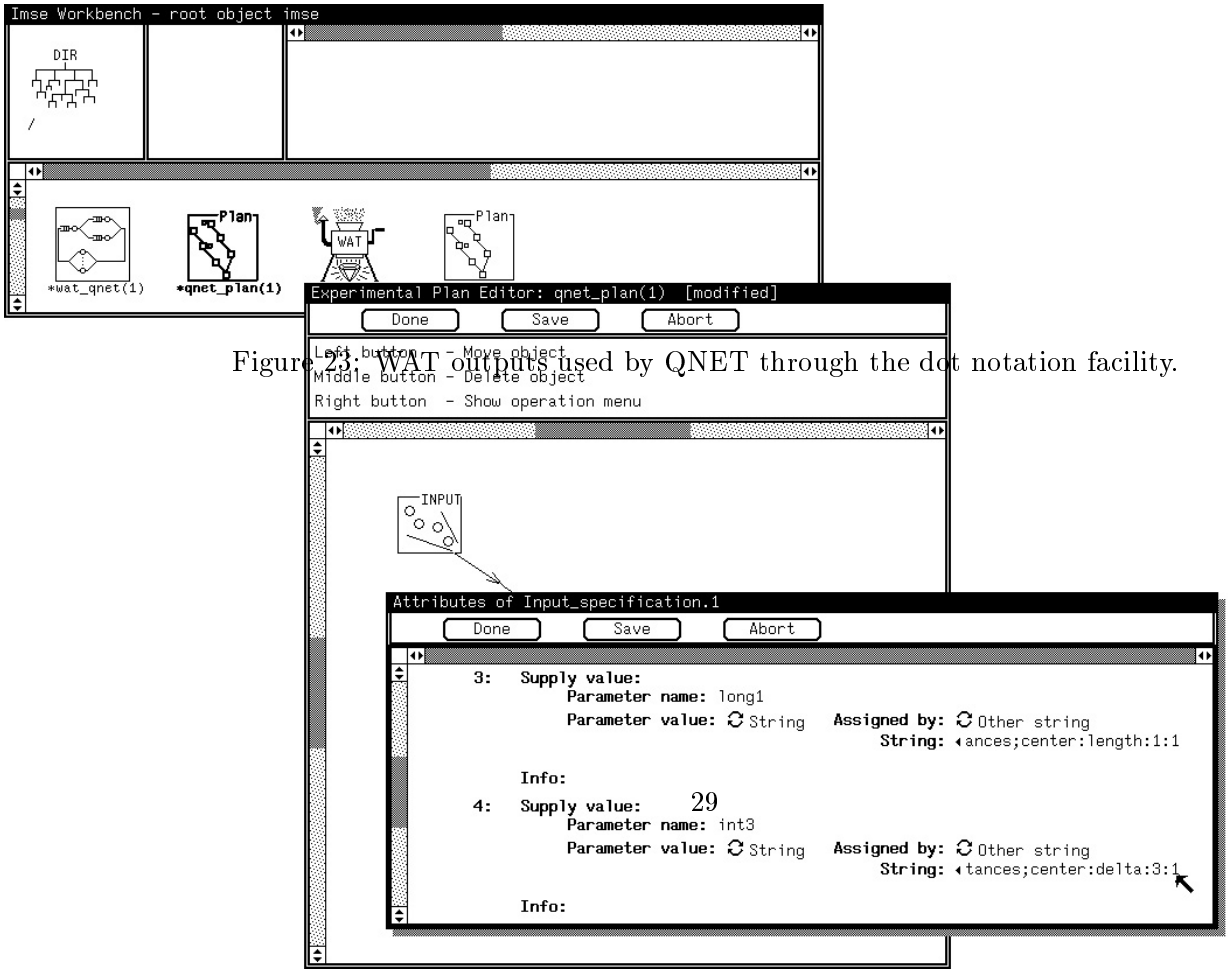


Figure 23: WAT outputs used by QNET through the dot notation facility.

APPENDIX

A Use of the components of the interface

The interface is a set of basic components defined in the SDMF tool [11]. To facilitate the use of this interface, we will briefly review the various components.

A.1 Buttons

There are two different kinds of buttons within the interface:

1. action buttons
2. subform buttons

Both of them can be activated by simply clicking the left mouse button with the mouse cursor located on the image of the button (i.e., the square surrounded label).

Figure 24: Examples of action buttons (`Done`, `Save` or `Abort`) and subform buttons (`Help` or `List format library`).

A.2 Menus

There are different types of menu within the interface which can be invoked and used in the same way. The following figures show a few examples of the menus provided within the interface.

Figure 25: Example of a window banner menu.

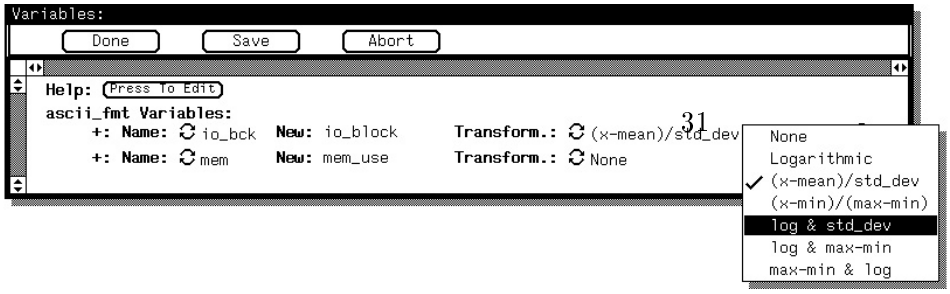
Figure 26: Example of a `FREE variable` menu.

To open a menu (that is, to display all the items it consists of), the user needs to press the right mouse button with the mouse cursor located on the image of the menu (e.g., the banner of the window, the sum item or the string item).
To select an item, the user, with the mouse button pressed, needs to drag the mouse cursor to locate it on the image of the item. When the mouse button is released, the item is selected and the action associated to it executed.

A.3 Sum items

Sum items are compositions of buttons and menus and can be used in either one or the other "mode".

Figure 27: Example of a sum item menu.



Using the "button mode", the user will change the value of the sum item, by selecting the value of the sum item immediately after the current one. A few clicks will allow the user to scan sequentially the list of values one by one.
Using the "menu mode", the user will display the list of values of the sum item and directly select one.

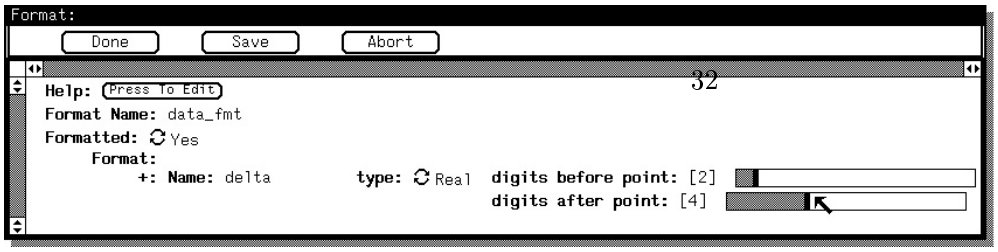
Figure 28: Example of a sum item.

A.4 Integer range items

Integer range items are composed of two parts: an information part which displays the current value of the integer range item and a slider which is used to change the value itself.

Figure 29: Example of an integer range item.

It is possible to move the slider to a specific position by clicking the left mouse button with the mouse cursor located in the specified position within the slider rectangle. The user can also make the cursor sliding by pressing the left mouse button with the mouse cursor located on the black cursor of the slider and dragging the mouse to the left or to the right hand side. While sliding the slider cursor, the changes of the values of the integer range item are displayed. The value is set when the mouse button is released.
When the mouse cursor goes out of the slider rectangle, the slider cursor returns to its original position.



A.5 String items

A string item is a field containing either integer, real or textual values specified from the keyboard.

Figure 30: Example of a string item.

When the user wants to enter a value, he just needs to position the mouse cursor on the field he wants to edit and then he can type the value.
Each field has a predefined type associated to it. A test on the type is performed to avoid possible errors (e.g., typing a character in an integer field).
The `DELETE` key placed on the keyboard is the cancellation character.

A.6 List items

List items are the building blocks of other types of item. This means that an element of a list item can be a composition of any of the other types of item.
List items can be displayed in two different formats depending on the initial number of elements in the list (see Figs. 31 and 32).

Figure 31: Example of a list item with no initial element.

In the first case (see Fig. 31), there are no initial elements in the list, the user only needs to click on the `Extend List` button to add a new element to the list.
In the second case (see Fig. 32), one or more elements already exist, by clicking on the "**+**"

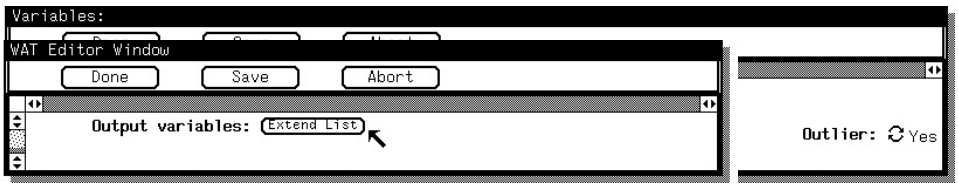


Figure 32: Example of a list item with initial elements.

button at the left hand side of any element of the list with the right mouse button a menu is displayed (see Fig. 33). It is possible to:

- insert a new element (just before the current one);
- append a new element (just after the current one);
- delete the current element of the list.

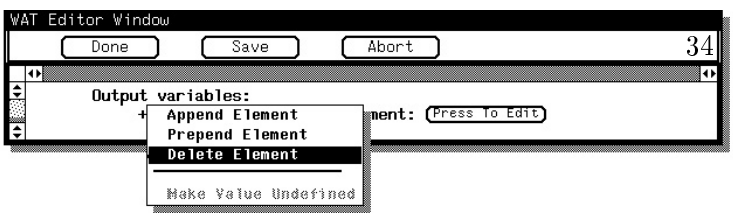
When the last element of the list is deleted, an empty list is obtained (see Fig. 31).

Figure 33: Example of the menu of a list item.

A.7 Undefined values

Making values undefined is a feature supplied by the SDMF through a special type of variables, called FREE.

This facility gives the user the possibility to postpone the setting of the values of a few parameters just before the execution, that is at a later stage than the WAT editing, e.g., within the Experimental Plan Editor. This allows the user to run compound experiments with the same general specifications, and to exchange data among the tools through the dot notation facility (see Sect. 6).



In order to leave a value not defined (where it is possible) the user has to click the right mouse button with the mouse cursor located on the selected item. A menu showing `Make Value Undefined` item will appear. Once selected, it will display the "`Unknown:`<label>" item (see Fig. 34). Note that <`label`> is a string item, acting as an identifier for the Experimenter.

Figure 34: Examples of different uses of `FREE variables`.

The `Unknown` item has a menu associated to its label, in which the user can select the `Edit Value Info` option, which displays an editor window (see Fig. 35) where additional information or comments can be added. These information, which provide some explanation about the use of the parameter, can be displayed either in the `INPUT` or `OUTPUT` nodes of the Plan.

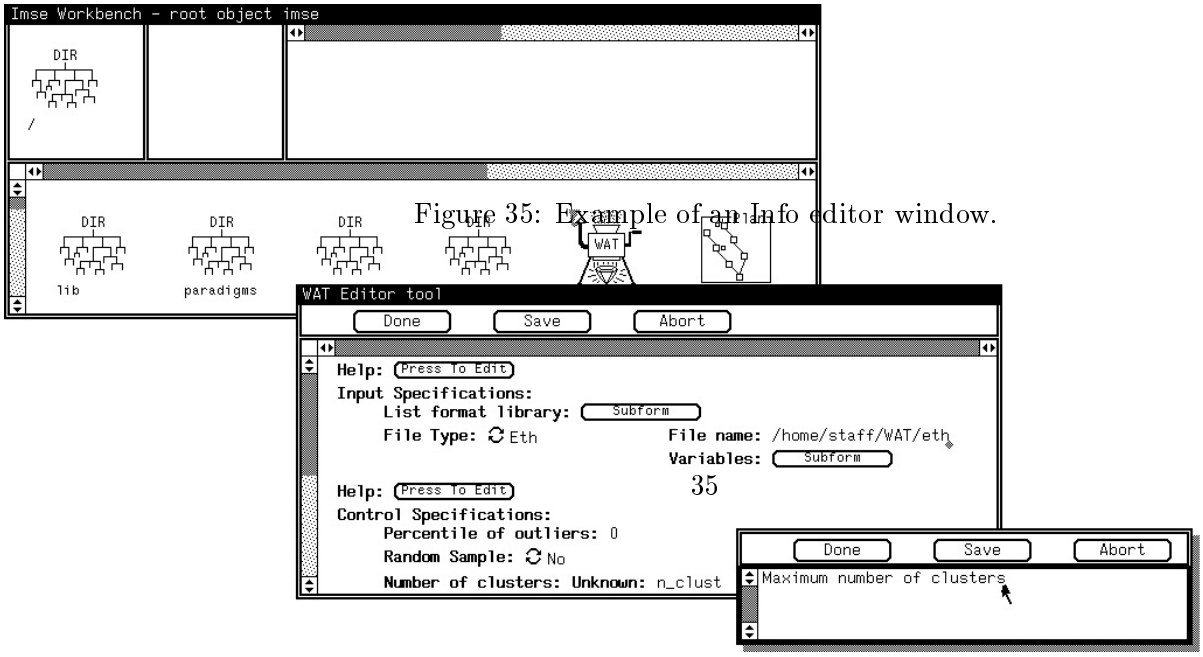


Figure 35: Example of an Info editor window.

A.8 Resizing a window

In many cases, especially when list items are used, it is necessary to resize the window in order to completely display its contents. This operation can be performed in various ways. The standard windowing procedures are available. An alternative simple approach is to use the item `Fit contents` in the menu of the window banner (submenu `Resize`, see Fig. 25), which will automatically resize the window according to its contents.