CNR - National Research Council
Istituto di Scienze e Tecnologie dell'Informazione
"A.Faedo"
Via G. Moruzzi, 1
56124 PISA

COW_Suite

User Guide

PISAtel Laboratory
ISTI – CNR

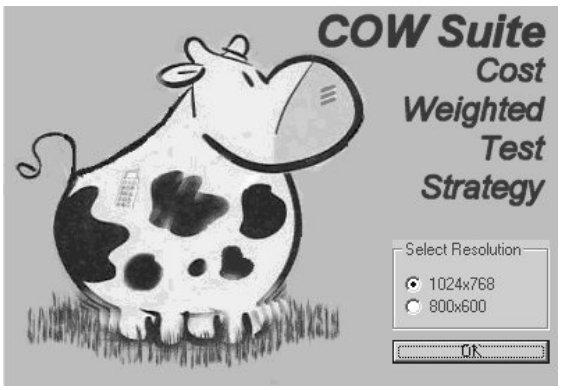
Contents

Introduction

CowSuite (COWtest pluS UIT) consist mainly into two component: the Cowtest strategy (Cost Weighted Test Strategy) component, which is in charge of test strategy selection and planning and UIT (Use Interaction Test) component which develops the test derivation. CowSuite is a tool for generating a suitable set of test cases based on UML language.

This tool analyses and processes a visual representation of software model getting information from files created by a specific tool UML design definition. The current prototype CowSuite version 1.0, processes ".mdl" files using Rational Rose tool [product informations at *www.rational.com* or directly at IBM new site : *www.306.ibm.com/software/rational*].

Rational Rose is a tool for developing UML design, which allows the generation of classes, components and sequence diagrams etc. CowSuite uses Rational Extensibility Interface (REI) and the COM library at TypeLibrary level. It refers this library as objects library in CowSuite project environment.



(picture 1)

NOTE The current version of CowSuite 1.0 accepts only REI version included in 2002.05.20 Rational Rose and does NOT interact with Rational Rose RT. We are going to implement RoseRT library interface

in order to allow ".rmdl" files processing, which are typical of RT version.[1].

CowSuite tool provides two screen resolutions (pixel setting):
- full screen 1024x768 pixel
- 800x600 pixel

User can set both using the initial window appearing at the starting of Cowsuite program (see picture 1).

This User's Guide has the purpose of helping the user in understanding the technical details and interaction with CowSuite program version1.0 (For more explanations about CowSuite approach theory see the Power Point documentation available on Pisatel Web Page (http://www1.isti.cnr.it/ERI/cowsuite/). It includes this User guide and the setup program to install CowSuite in the PC).

Following this guide the user can understand the steps necessary during the project analysis and design (realised by Rational Rose), for increasing the performance of CowSuite tool such as: the refinement requires to the project design or the definition of specific conditions for better guiding the automatic derivation of test cases.

Installation requests

As mentioned in the Introduction, Rational Rose 2002.50.20, or following versions, needs to be installed before CowSuite 1.0 version. If a Rational Rose RT version is already installed on the PC, CowSuite is not able to work correctly. In this case it is necessary that the user sets the TypeLibrary.

If both Rose e RoseRT have been installed in the system, the TypeLibrary reference works correctly only if Rose is the last application used before starting CowSuite.

Finally for making sure that CowSuite behaves correctly, the American numeric format must be set in the host Window system (number format i.e. "." as decimal separator, and "," as thousands separator).

For this operation user must open the *"Control Panel"* window, and select "*Regional Settings*" option. These settings should vary for the diverse Windows systems (Windows 98, Windows 2000, Windows XP

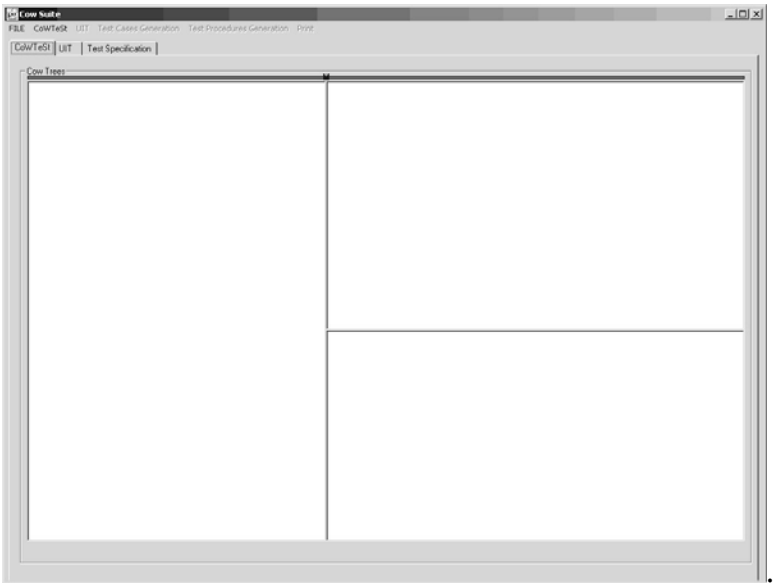
[1] REI libraries used in both the program versions are different in some calls to internal methods: in Rose version methods are completely documented using some names and an object structure dedicated to diagrams use; while in the RoseRT even though it includes most of the methods used in Rose version, the library structure is completely dedicated to the components and their development and management. Thus it can't be used within the current version of CowSuite program.

and so on). For details see the operating system's reference book of your computer.

CowSuite tool

The Cow_Suite consists of three work phases corresponding to three different units: CoWTeSt, UIT e TestSpecification (see picture 2), displayed up on the left, under "File*"*..

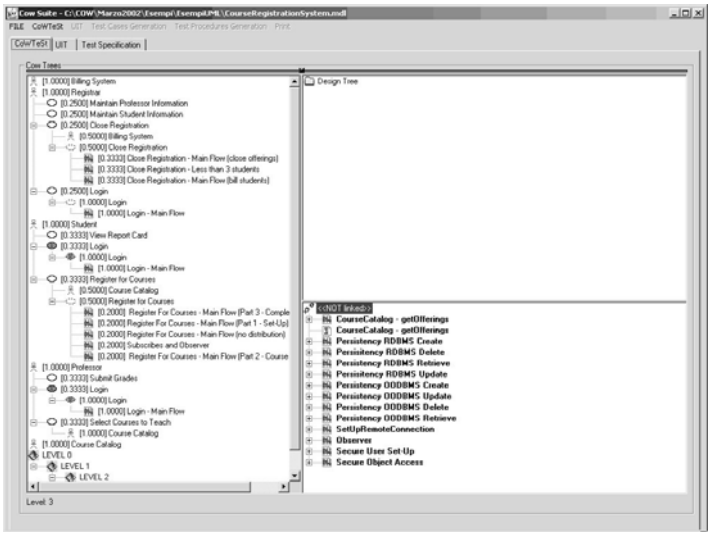
Note that, in order to use correctly the tool, the three phases must be started in the same order in which they are displayed..



(picture 2)

User can set the internal area size of the windows using the horizontal scroll bar

In the first phase (CowTest) the user must enter the name of MDL file, using "`FILE -> Load MDL File`". CowSuite picks up from this file the structures of the UML elements (Use Cases, Actors, Sequence Diagrams and so on) useful for test case derivation (Note: the tool has been developed for UML1.4). The elaboration then proceeds with the construction of a set of trees as shown on the left area of picture 3.



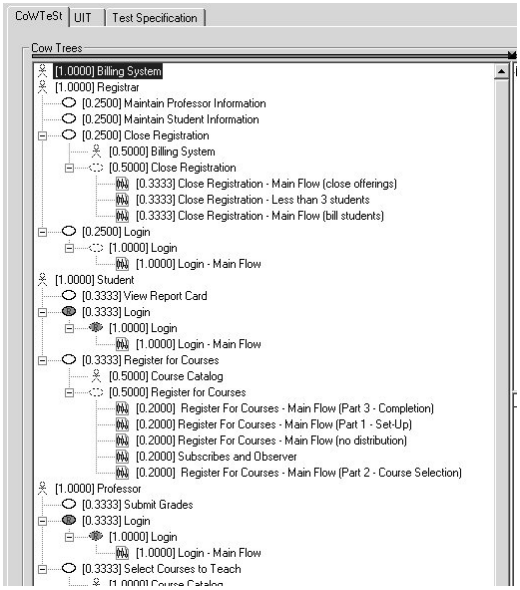
(picture 3)

In the top-right area, the packages, as they have been created in Rational Rose environment, are showed (Design Tree) while in the bottom-right area, the Sequence Diagrams not linked to a specific Use Case and/or the wrongly linked elements are listed.

For knowing how to correctly link a Sequence Diagram to its own Use Case see chapter "Using Rational Rose".

In order to organize all the elements in a hierarchical structure, the tool selects from the Use Case Diagrams included in the ".mdl" file, first the Actors and the Use Cases and their corresponding relationships. These elements are structured in several trees having these peculiarities:

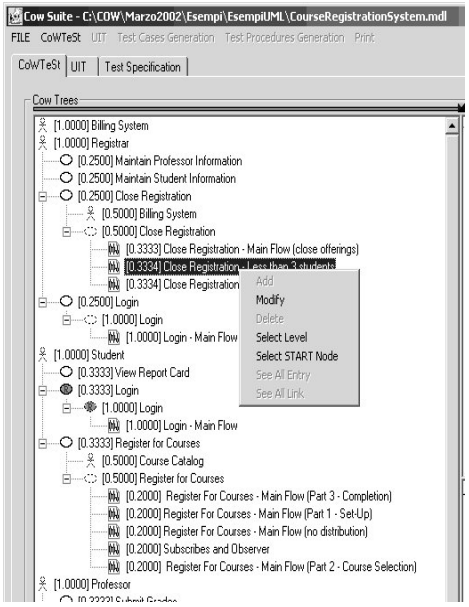
- The root is always represented by an Actor who represents the interaction between external environment and the system.
- Second level nodes are always Use Cases, which describe in details the functionalities of the systems.



(picture 4)

All the generated trees use the same Rational Rose graphic representation (Icons, Symbology) for the relative nodes. Specifically the name of each node is the same referred by the objects included in Rose MDL file. Thus this is only a different structural view of the elements already included in the UML design created with the Rational Rose environment.

CowSuite picks up and then displays the information retrieved from the Rose file.



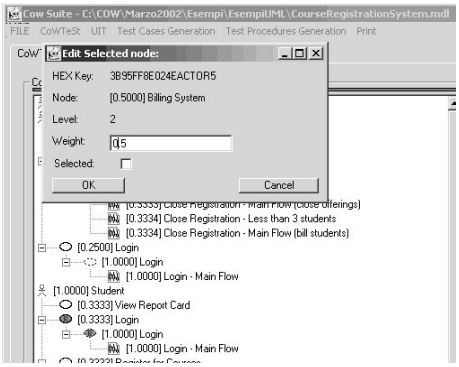
(picture 5)

Each tree is composed of several levels, named “integration levels”. The last row in left side tool window represents them in synthesis (user should use window scroll bar).

CowSuite automatically associates a real number, called “weight”, to each node of a specific tree. The range is [0.000, 1.000] considering that: 0.000 is the smallest, i.e. less important from a testing point of view and 1.000 is the highest. By default, the weights are distributed in a uniform manner so that the sum of the weights of the nodes at the same level in the tree is 1.000. The chapter “Using Rational Rose”, later on this guide will detail all the exceptions to this rule.

According with the importance of each level, user can modify the default weights, entering the proper values between [0.000,1.000].

To assign a new weight to a node, the user must select the node, open a scroll menu using mouse right key and select the “Modify” option. A dialog box appears and the user can modify the default value shown. As a consequence CowSuite will automatically update the weights for the sibling nodes belonging to the same level so that the total is still 1.000.



(picture 6)

NOTE CowSuite does not modify the previously modified values, but only the weights of the reminder nodes

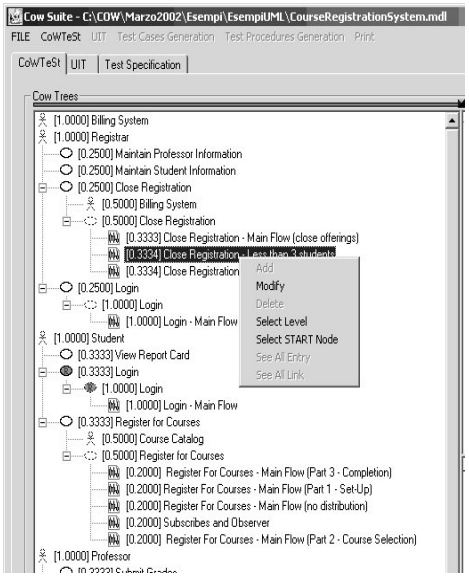
In case of the modification involves all nodes at the same level, the user must manually verifies that the inserted values have 1.000 as overall sum.

Hint: leave some nodes non-modified in order to allow CowSuite to automatically recalculate the remaining weighst and check their sum.

In this phase the derivation of test cases occurs considering one tree per time. Therefore the user first selects the tree of interest, and then proceeds as described in the following chapter.

Selecting a whole tree (from sublevel to Actor)

In order to select a whole tree starting from ROOT node (identified by an Actor Symbol) to a lower level, the user has to click with the mouse right-key (please refer the scroll menu appearing during the weight modification phase, see previous chapter) and then select **"Select Level"** option. The selected parts of the tree will be putted in evidence using a background colour different from the rest of the graphic structure.



(picture7)

NOTE: all the nodes belonging to the lower levels are not affected by the selection.

By means of this selection the program asks the user to select the proper test strategy to be implemented. The tool can implement two approaches:

- a fixed number of test cases to be distributed among the selected node in the tree (identified by Number of Test Procedures)
- a fixed percentage of functionalities to be covered by the derived set of test cases. (identified by Functional Coverage):

These strategies will minutely be describe in the following chapters.

Partial selection of a tree

To execute a partial selection of a tree it's necessary to perform the following steps:

1. select the node on the start level using the mouse right-key and click on the **"Select START Node"** option.
2. select the node at the end level and click on the **"Select END Node"** option.
3. In order to select definitively the marked part of the tree user has to click, using the mouse right-key again, on the "Select Level" option.

In particular:

1. Identify the start node of the selection; the node selected must be a branch node (it must have at least one child), otherwise an error message will occur. The selected node will appear with a red background.
2. Identify the end node of the same selection. Also this node will appear with a red background.
3. confirm the selection of the set

The program will highlight the parts of the selected set using a colour different from the rest of the tree and, at the same time, the labels with red background will disappear.
Through this selection the program activates, in an automatic way, the request of test strategies selection to the user. (for further details see also the next chapter):

- a fixed number of test cases to be distributed among the selected node in the tree (identified by Number of Test Procedures)
- a fixed percentage of functionalities to be covered by the derived set of test cases. (identified by Functional Coverage):

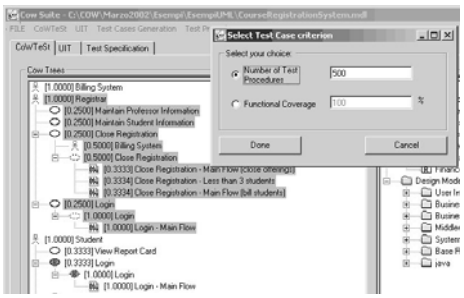
We will explain in detail the application of both the strategies in the following chapters.

Testing strategies

After the selection of the interested area (whole tree / tree subset) the program activates the request of test strategies that will be used for test cases generation. Cow_Suite can implement two test strategies:

1. Distribute a fixed of test cases (called in the rest of this document test procedures) to be distributed in the selected area: *Number of Test Procedures*

2. Define a fixed percentage of ercentage of functionalities to be covered by the derived set of test cases.: *Functional Coverage*.



(picture 8)

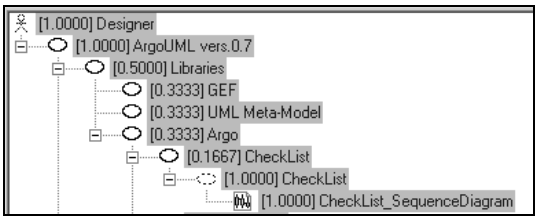
Picture 8 shows the dialog box in which the user can specify the number of test procedures or the percentage of functional coverage to be applied.

In the next chapters, further details about the interaction between the program and the user in both selected strategies are provided. In any case, all the nodes selected in this phase will be displayed in the left area (tab. UIT) during the second phase, named UIT phase. The description of the UIT methodology will be provided later in this guide.

Test strategy 1: Number of Test Procedures

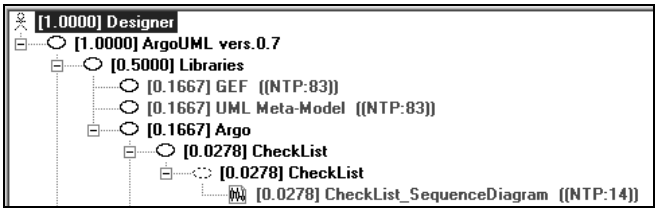
Selecting this distribution method, the selected number of Test Procedures is distributed among all the leaf nodes (or all the nodes belonging to lower levels, in case of sub-tree selection) of the selected area using their final weight (calculated as product of the weights from the root node to the relative node). In particular the program will assign, for each node, the specific number of Test procedures calculated multiplying the fixed number of Test Procedure by his final weight.

For instance: let us suppose that in the first phase “Creating Cow Tree” the user has associated to the node the weights distribution of picture 9, elected the entire tree entering and asked for 500 Test Procedures to be generated:



(picture 9)

the program will compute the final weight for each node and display, in the UIT window, the involved nodes with their **new** final weight calculated as product of the weights from the root node to the relative node (see below);

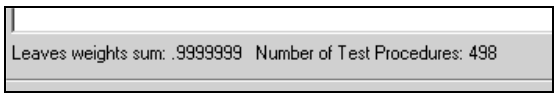


(picture 10)

In particular, the program will add, to each node, a label ([NTP:xx]) which indicates the number of assigned Test procedures to be generated for this specific element.

Taking in consideration also the possible rounding off, the sum of the assigned Test Procedures (total of NTP:xx values) will be the same entered by the user.

In the bottom of the window (area status) the program displays the total number of Test Procedures resulting after adding all the Tests calculated for each node. This value will be, as much as possible, closer to the value entered by user (picture 11).



(picture11)

Test strategy 2: Functional Coverage

If a fixed percentage of functional coverage is entered , the final weights calculated by the program for each node will be used in order to identify, in the selected area, the part of the selection (leaf nodes) to be considered. In this case the program indicates the smallest number of test procedures that will be necessary to reach the specified coverage. In any case the user can always modify the proposed number.

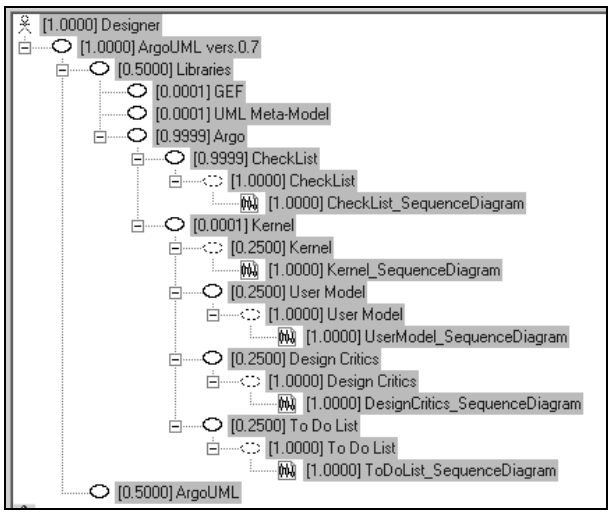
In particular, in order to perform the nodes selection the program :

- calculates for each node the final weight considering all the weight from the root node to the selected node;
- orders (as regards to final weight) all the nodes, considering the selected level of integration, and associate to each node a percentage value proportionated to the associated weight (a weight = 1.0000 meet a percentage = 100 and so on);
- selects the nodes in decreasing way, starting from the node having the highest final weight since their sum is greater or equal than the entered percentage value. The program puts in evidence the selected nodes with a green colour and the remainder nodes with a red colour.
- normalizes automatically the node weights so that they are still between 0.0000 and 1.0000 and the sum of weights the selected still remain 1
- calculates the lowest number of Test procedures needed for the requested percentage of coverage

Then the program proposes the lowest value of Test procedures needed for requested coverage and ask sthe user to confirm it.

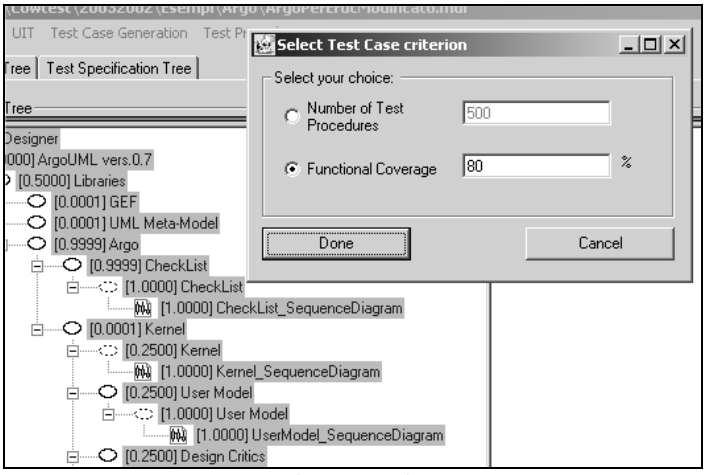
In this phase, the user can insert a greater number of Test procedures to be distributed among the selected nodes.

Let us suppose we have modified the weights in order to obtain a distribution like that in picture 12:



(picture 12)

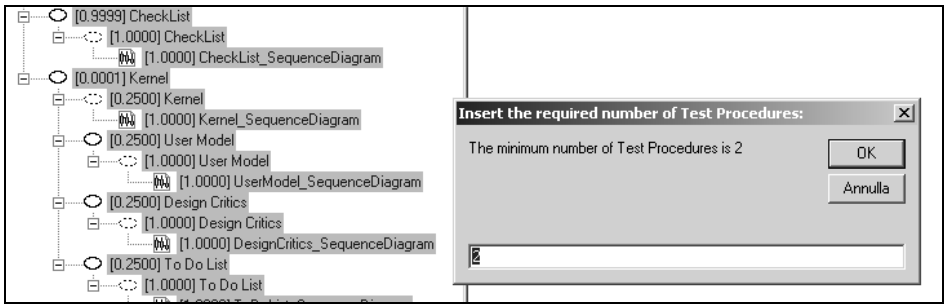
and then we have elected the Functional Coverage option and setted a percentage value equal than 80% of total.



(picture 13)

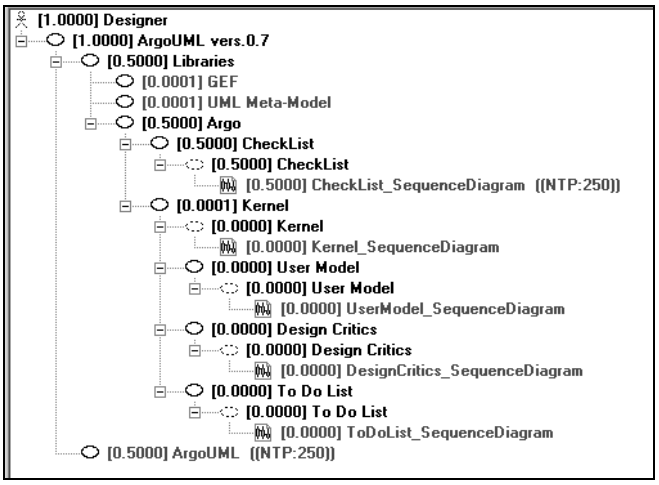
After the confirmation (by pushing the key “Done”) the program displays the lower value calculated. In this case the values will be 2

because the program has selected a Test Procedure for "**[0.5000] ArgoUML**" node, and another one for "**[1.0000] CheckList_SequenceDiagram**" node.



(picture14)

Now the user can set the number of procedures, 500 for instance, and confirm the procedure start. The following picture shows the consequent distribution:



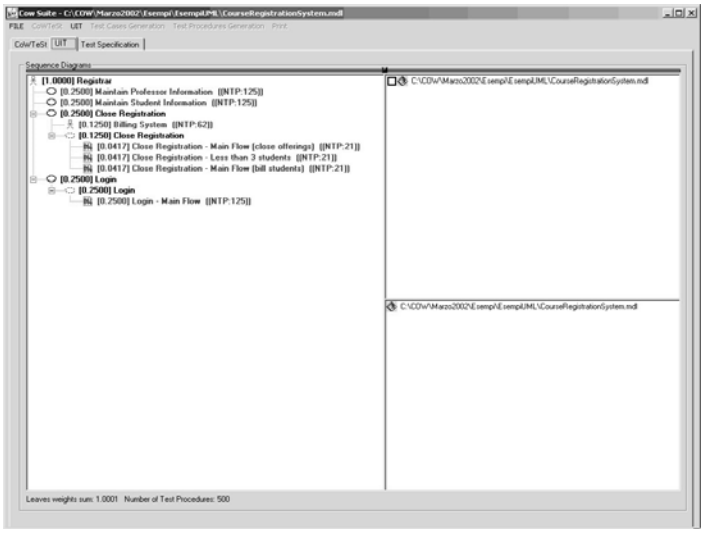
(picture 15)

We can verify that::

- the distribution foresees 250 Test Procedures among the nodes "**[0.5000]ArgoUML**" and "**[1.0000]CheckList_SequenceDiagram**" respectively;
- the program has set to 0.000 the weight of the nodes not involved in the calculation

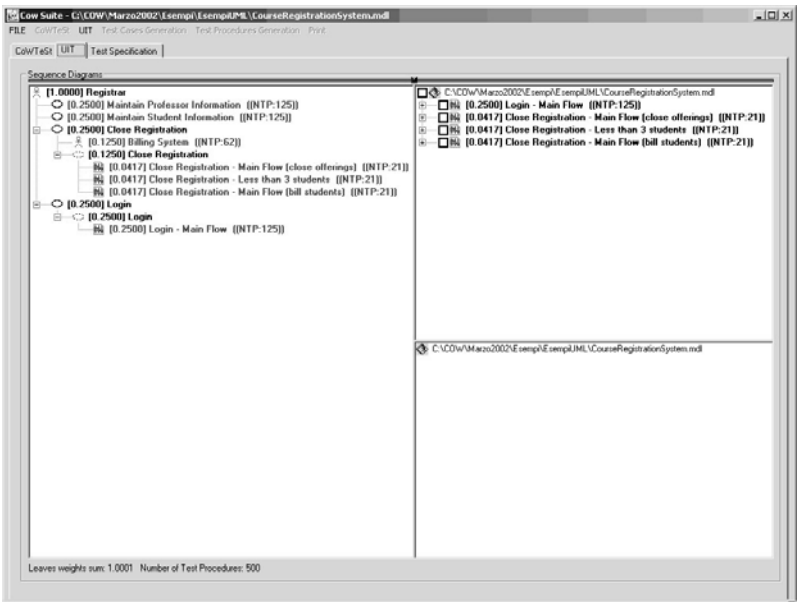
Second phase (UIT)

Picture 16 shows what the user get after selecting the UIT option. In the left area the program shows the copy of the tree selected in the previous phase. There main difference between these two views relies on the fact that the final weights have been calculated and the assigned Test Procedures for each node displayed on the left of each node (NTP=xxx notation)



(picture 16)

In this phase, the program activates the UIT menu that can be used for the selection of Sequence Diagrams set. This set will be handled in the next phase and will be used in order to create the Test Procedures lists (see below)

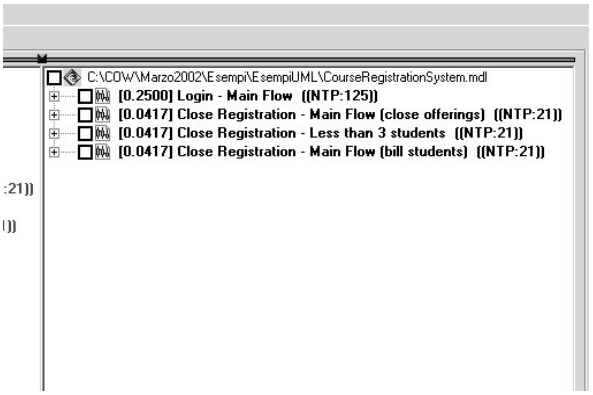


(picture 17)

In order to define the test procedures, the user must select **"Loading from Cow Tree"** option. Afterwards the program takes the information from the structure selected in the first Cowtest phase.

So the program loads in the right-up area all the Sequence Diagrams, or Collaboration Diagram, included in the selected structure. The final tests definition occurs in two different steps: (Test Case definition and Test Procedure definition) only considering the Sequence Diagrams and Collaboration Diagrams included in the selected tree.

In particular the program will , in order to generate the final list of Test Procedures, fist produce the structures of the Test procedures (during the test specification phase) called the Test Case set; then by iteracting with the uset the program will fill them with the proper values obtaining the final test procedures set.

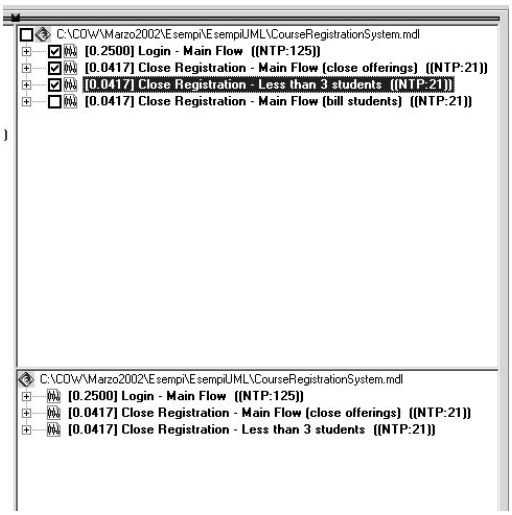


(picture 18)

Once the list of the available Sequence/collaboration diagrams is displayed, the user can freely operate the diagram selection. All the

selected diagrams will be displayed in the right-down area (see picture 18).

NOTE: The test case generation will consider only the Sequence/Collaboration Diagrams moved in the right-down area. For instance, in order to move the first three Sequence Diagrams in the right-down area, the user must select the interested nodes and move them, pushing the mouse left-key, in the right-down area. Once he is in the area, he musts release the key so the program will add them to the list (see following picture).

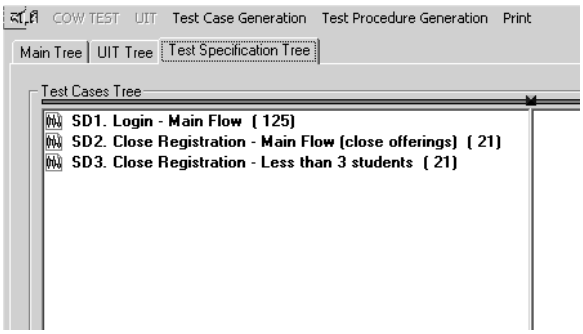


(picture 19)

Note that each Sequence/collaboration Diagram added to the right-down area, will remain selected in the right-up area in order to avoid that the same diagram should will be moved several times in the set. If the user tries to move more than once the same Sequence/Collaboration Diagram, the program will arise an error message and will refuse the operation.

Third phase (TestSpecification)

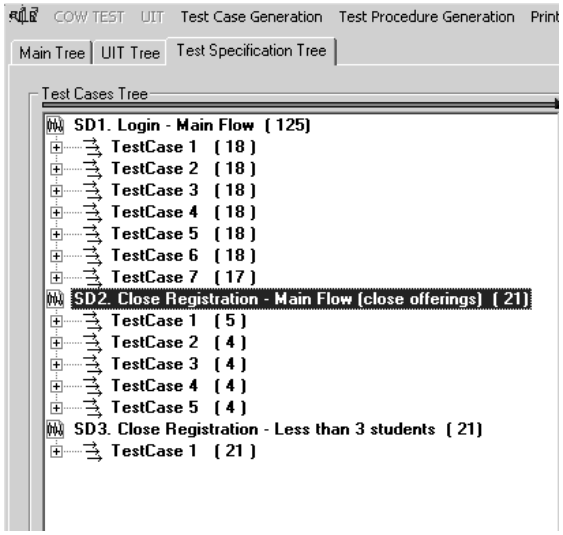
The test specification phase is divided into two steps: the first step in which test Case list is defined and the second step in which program generates the test procedures..



(picture 20)

In the first step , the program will apply the UIT method in order to associate the list of derived test cases to each Sequenze/Collaboration Diagram selected, using the information picked up from MDL file.

In particular in order to display the test cases of each Sequence/Collaboration Diagram, the user must activate "Test Specifications Tree" option by clicking on "Test Case Generation" and then on "Processing" option.



(picture 21)

The program displays, for each Sequence/Collaboration Diagram, the derived test case list using UIT method. Considering picture 21, the number between square brackets, displayed on the right of the name of each Sequence/collaboration Diagram, is the number of test Procedures assigned to this diagram in the previous phase. This number will be uniformly distributed among the test cases associated to the diagram.

A double click on each Sequence/Collaboration Diagrams activate the display of the associated test cases list .

On the right to each Test Case the user can see the number of Test Procedures to be generated using the test schema described by the same test case.

If the user selects a test case, the program will show, in the right-area, its internal structure: the list of values used by each parameter and the sequence of Test case procedures.

If the user performs a double click on the Test Case, the program will open the left tree and will display more nodes and all the Test Case call sequences.

Therefore the program shows the procedures included in the associated Sequence Diagram and set up by Rational Rose (MDL file).

In the left displayed area, operating in opening the nodes, the program will also show the possible parameters belonging to each method of each class (Rose object).

Test Procedures generation

The next step is instancing test case for Test Procedure generation. In activating “Test Procedures Generation” menu, the user can create all the possible Test Cases (“All Test Case” option) and show them, in the right area , using “Visualize Test Scripts” option.

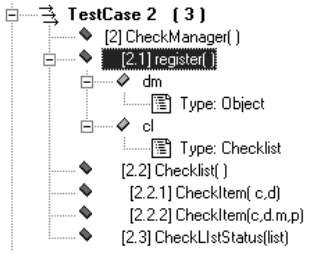
For Test Procedure definition, the user has to interact with the program setting suitable parameters. If the number of values is not suitable for generating Test Procedures, the program will display a warning message like this:“`WARNING!!! No values for Test Procedure generation in Test Case n`”.

For setting the values , the user has to operate in the left area, in which the program displays Test Case and each case parameters took using even class diagram info. After pushing on the mouse right-key, the program will open “Specifications” page of related parameters in which the user can define:

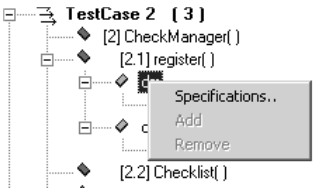
- default values setted by Rational Rose (see next charter);
- user values;

Default values and User values

In picture 22a we draw attention to a method "`register()`" with two parameters "`dm`" and "`cl`". After selecting a parameter and pushing a mouse right-key, the program displays a sub-menu useful for opening parameters specification (picture 22b).



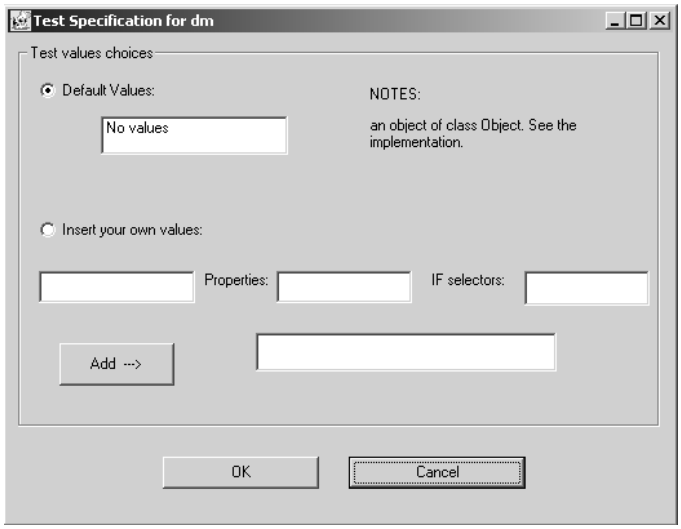
(picture22a)



(picture 22b)

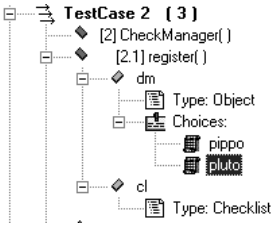
Selecting "`Specifications`" option, the user can confirm default values (if present) or insert one's values to be used in test case generation.

Picture 23 shows the window used for parameter values management:



(picture 23)

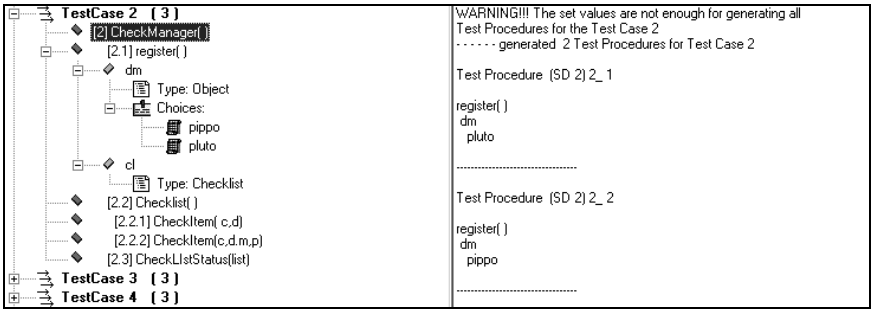
Possible default values (set by Rational Rose) are shown in the box nearby "`Default Values`" , whereas user values must be entered using "`Add ->`" key in "`Insert your own values:`" option



(picture 24)

Let us suppose that the user has introduced two values, "pippo" and "pluto" for instance, the program will place them in "`choices`" associated to the selected parameter (picture 24).

In order to allow that these parameters will be used for test procedures generation, the user has to repeat the generation procedures "All Test Case"and "Visualize Test Scripts" (see picture 2):

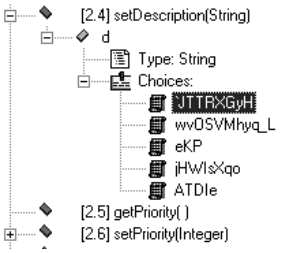


(picture 25)

Note the warning message about the impossibility of generating all the Test Procedure for Test Case 2.

If the parameter contains default values, the user can decide to use values selecting Specifications" option and pushing "OK" without changing anything.

Note: in this release of the program, it is necessary to activate the display of each parameter, even for enabling default values.



(picture 26)

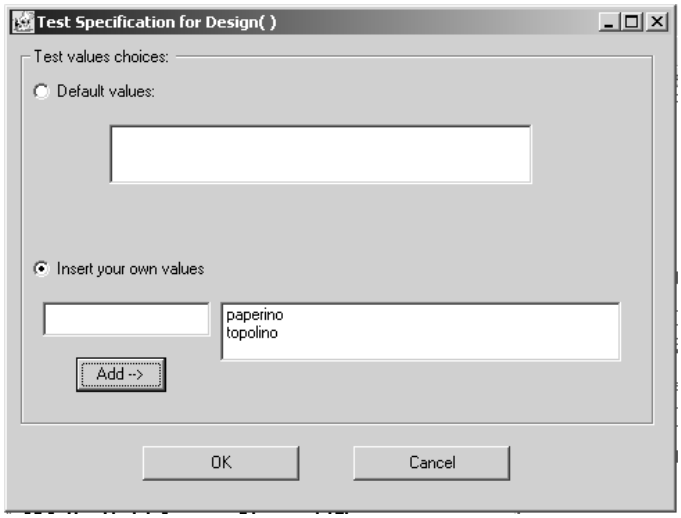
Picture 26 shows an example of default values for the parameter “d”, type “`String`”, as quoted in the related call to “`setDescription(String)`” method.

To do this action, the user has to select “d” parameter, activate, by pushing mouse right-key, “`Specifications`” option, and then ,after verifying that default values are displayed in bottom area , select “OK”.

Method default values and user values

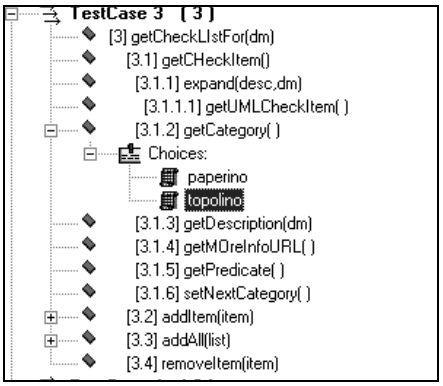
Similarly, the user can set default values for each method.

After selecting the interested method and “Specifications” options, by pushing a mouse right-key, the following window will appear:



(picture 27)

After clicking on “OK” button, the program will put in the left area the values formerly setted as method “choices”:



(picture 28)

Following up every values update we need to execute again "`All Test Case`" and "`Visualize Test Scripts`" generation procedure in order to obtain again the generation of output Test Scripts .

Program output

In the actual CowSuite version the output files are text files generated in the same directory of MDL file created by Rational Rose in temporary mode. *Therefore when the program CowSuite ends all the following files are automatically deleted.*

The format of temporary files is :

1. SDn1<sequence Diagram name>(NTP1)TestCase(n2)(NTP2).txt
2. TestScript n3.txt

In which:

n1: is the sequence number of the relative Sequence Diagram;
<Nome del sequence Diagram>: is SD name;
NTP1: is the number of SD Test Procedures ;
n2: is the sequence number of SD TestCase ;
NTP2: is the number of TC Test Procedures;
.txt: file type (extension);
n3: is the sequence number of the generated TestSript

Type 1 files will include, for each Test Case, the sequenze of related operations and they are like:

```
Test Case  1   of  SD Kernel_SequenceDiagram



 Description:

 PreCondition:

 Flow of Event:
            Design( )
            DesignMaterial( )

 Categories

     Settings Categories :

     Interactions Categories :
                  Design( )
                  DesignMaterial( )

 PostCondition:

 Comment:
```

Whereas type 2 files will include all parameters set and the procedures to be applied to the entry model.

They are like:

```
Test Procedures generation file of  ToDoList_SequenceDiagram


WARNING!!! No values for Test Procedure generation in Test Case
1


WARNING!!! The set values are not enough for generating all
Test Procedures for the Test Case 2
- - - - - - generated  5 Test Procedures for Test Case 2

Test Procedure  (SD 2) 2_ 1

setHeadLine(String)
 h
   ]

---------------------------------

Test Procedure  (SD 2) 2_ 2

setHeadLine(String)
 h
   D^eamqH`al
```

```
---------------------------------

Test Procedure  (SD 2) 2_ 3

setHeadLine(String)
 h
   qXiE[pj^HRRPVFw

---------------------------------

Test Procedure  (SD 2) 2_ 4

setHeadLine(String)
 h
   AfyinQDAtOL

---------------------------------

Test Procedure  (SD 2) 2_ 5

setHeadLine(String)
 h
   GsQFuKWDCn_
```

Rational Rose usage

This chapter briefly describes how the user can insert some information directly into Rational Rose model and then which is the proper format so that the tool is able to read and load them directly in COW session.

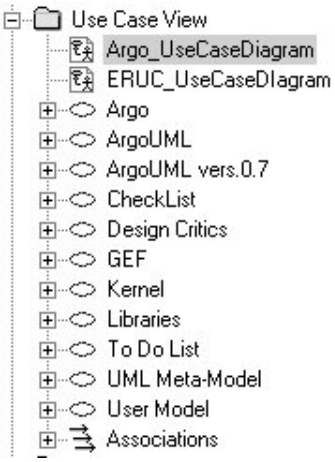
In detail we will discuss following topics:

1) How to link Sequence Diagrams to a Use Case;
2) How to define Use Case weight;
3) How to insert attribute default values ;
4) How to insert Iteration Categories default values;
5) How to insert IF ELSE cases to the messages in order to modify test procedures

Points 2, 3 and 4 are optional, since the user can do these operations even inside the tool; whereas point 1 is necessary in order to implement COW Tree for test cases generation.

Sequence Diagrams link

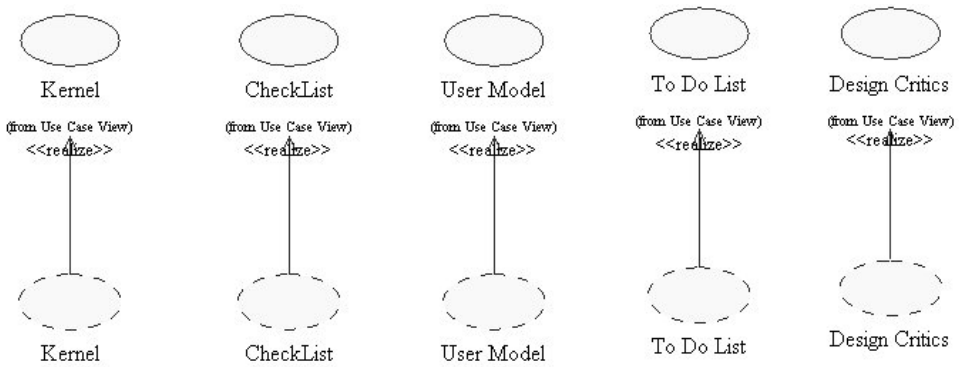
During the Rational Rose model creation, the user puts in “Use Case View” package all the Use Case and all the Use Case Diagrams implementing Use Case relationships.



(picture 29)

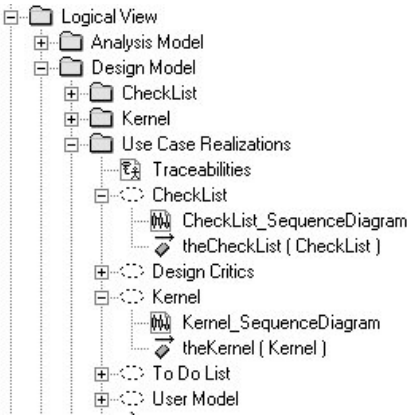
In order to link Sequence/Collaboration Diagrams to an Use Case the user should proceed as fallows:

- In creating under “Logical View” package, a package named “Design Model” and inside this one, another package named “Use Case Realizations”;
- In creating, inside “Use Case Realizations”, a “Use Case Diagram” named “Traceabilities” in which he have to implement a new Use Case, having the Stereotype set to “use-case realization” and the same name of the Use Case he what to link to.
- for each Use Case created, the user must create a new Use Case referred to the interested Use Case meaning that Rose have to pick up this Use case from Use Case View.
- Finally, the user must create an association, having the Stereotype set to “realize”, between the new Use Case and the linked Use case. Picture 30 shows the links (<<realize>>) for Use Cases: Kernel, CheckList, User Model, To Do List e Design Critics with Use Case included in Use Case View.



(picture 30)

Now, the user can see under “Use Case Realisations” package, all the created Use Case (hatched) in which he has to insert related sequence diagrams.



(picture 31)

Picture 31 shows a sequence diagram (kernel e CheckList) correctly associated to related Use Case. In this way, CowSuite tool is able to create the tree including all sequence diagrams and their related nodes.

Use Case weight

As mentioned above, Cow_Suite associates to each node for the tree, created from Rational Rose UML model, a weight between 0 and 1.0 so that the total weight at the same level is 1.

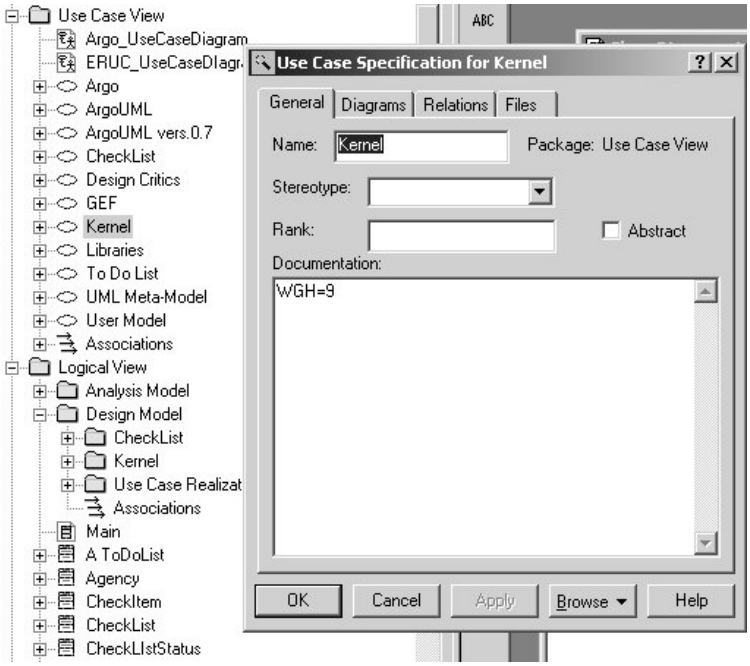
In this operation Cow_Sute will take in consideration the user preferences expressed in as an absolute importance factor for each Use case inside the UML design. The absolute importace factor for a Use Case is a value defined between 1 and 9 and inserted using the object Documentation.

When this data are available Cow-Suite during the weights distribution does not provide the uniform distribution for the involved UC as described in the previous sections but il will take in consideration also the absolute relevance of each UC. If no specification is included in Use Case Documentation, the tool will assign the value 5 as default weight that leads to a uniform distribution for the Use Cases waights.

Note that: picture 32 shows how the user can set Use Case weights in Use Case View. Once opened Use Case Specification for the selected use case insert in the Documentation field the relative value using the label:

WGH=*value*

Value must be a number between 1 and 9 (1 is the lowest value and 9 is the highest).



(picture32)

Parameters default values

The user can associate, directly in Rose model, the default values for each parameter to be used in test procedure generation, similarly to Use Case weight association (see previous charter).

All the messages belonging Sequence Diagrams are analyzed evaluating related parameters and each test case is generated using right values for these parameters.

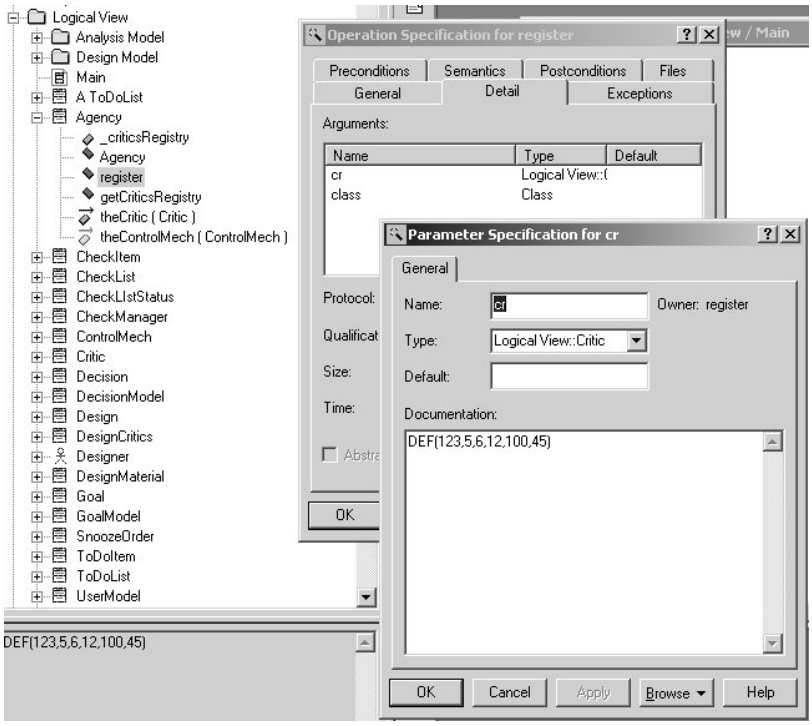
CowSuite is able to accept parameters values needed for covering all test cases by picking them up by UML model or by user interface.

If the user wants to insert the parameter default values , he should:

- Identify the message class;
- Open the message Specification;
- Show the message details in which he can see all the parameters;
- Open parameter Specification;
- Insert the values in Documentation field like:

DEF(1,2,3,4)

In Picture 33 you can see how the values 123, 5, 6, 12,100 e 45 for “cr” parameter that is the first argument for the message “register” of “Agency” class can be defined.



(picture 33)

How to insert IF ELSE to the messages for modify test procedure

Test case generation considers all the conditions included in UML model.

Note that: These conditions can be inserted only by rational Rose and NOT by CowSuite tool.

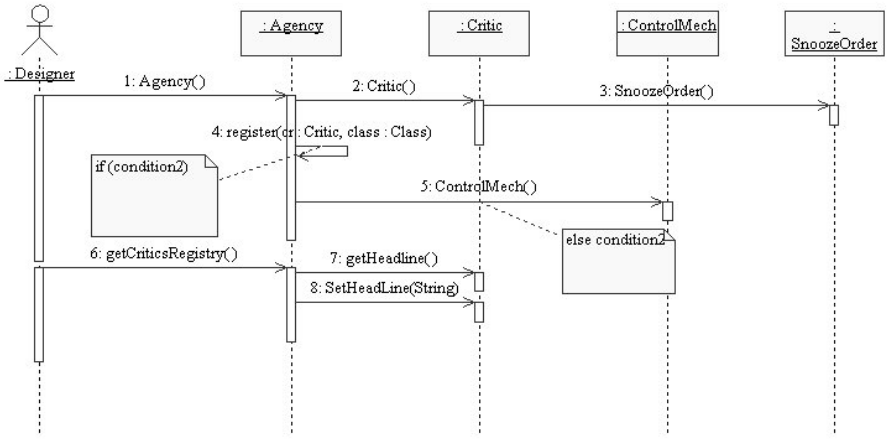
In order to insert these conditions the user should operate on sequence diagram messages

By using “Note” objects you can associate the condition to the message.

Possibile conditions are:

- if (condition)
- else condition

IF and ELSE must be capital letters.



(picture 34)

Inside Rational Rose, the user can specify the conditions by message tab Precondition.

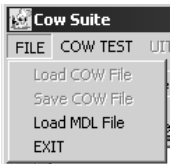
In this case the user must open message Property and specify the condition in Documentation filed of tab Precondition using the same syntax.

Menu description

This chapter shows the description of all the menu items included in the current version of CowSuite tool.

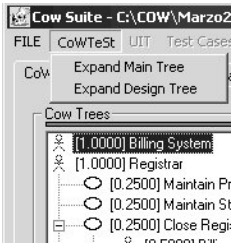
Menu items that appear disables will be implemented in the future versions.

Menu FILE



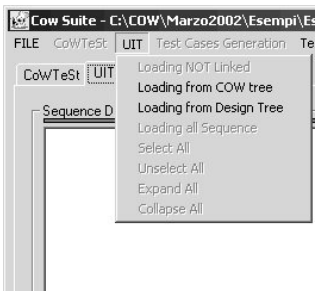
`Load MDL File`	Load and select a Rational Rose .MDL file
`EXIT`	Exit from the program and delete all temporary files

Menu CoWTeSt



`Expand Main Tree`	Expand the whole nodes of the tree realised in the left-screen side
`Expand Design Tree`	Expand the whole nodes of the tree realised in the right-screen side

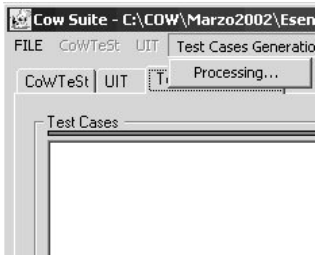
Menu UIT



`Loading from COW tree` Load the sequence diagrams from the tree generated by COW; SDs are shown in right-screen side after a subsequent selection.

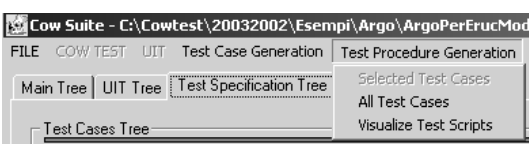
`Loading from Design Tree` Load the sequence diagram from Design Tree; SDs are shown in right-screen side after a subsequent selection.

Menu Test Cases Generation



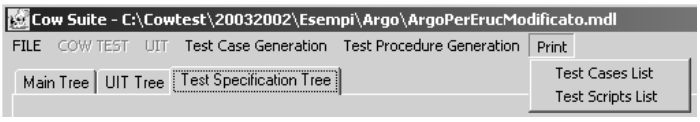
`Processing...` Start Test Case generation process; it picks up the test cases by Sequence Diagrams list and shows them in low-right side from UIT screen .

Menu Test Procedures Generation



`All Test Cases`	Generate all the Test Case
`Visualize Test Scripts`	Show generated Test scripts.

Menu Print



`Test Cases List`	Allow to print , in the system default destination, the list of all the generated Test Cases, before deleting them.
`Test Scripts List`	Allow to print, in the system default destination, the list of all the generated Test Scripts, before deleting them