

## THESIS / THÈSE

### MASTER IN COMPUTER SCIENCE

#### Critical study of an usability inspection method: the cognitive walkthrough

Grosjean, Christophe; Marin, Samuel

*Award date:*  
2000

[Link to publication](#)

#### General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

#### Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.



---

FUNDP

INSTITUT D'INFORMATIQUE

CRITICAL STUDY OF AN USABILITY  
INSPECTION METHOD :  
THE COGNITIVE WALKTHROUGH

Christophe Grosjean

Samuel Marin

Thesis submitted in the fulfilment of the requirements  
for the degree of

Master in Computer Science

Rue Grandgagnage, 21 • B – 5000 Namur (Belgium)

1999-2000

## **Abstract**

Finding out usability problems is always a key function in the design process of software. There exists a lot of assessment methods who can permit to evaluate an interface. Usability inspection-based methods are methods which can be performed without user.

The object of our work is to present in details one of those usability methods : The Cognitive Walkthrough (CW) method. The first main point we will give is a detailed explanation about this method and a theoretical foundation of it. Then we will place this method in the lifecycle of interfaces design process. Finally, we will specify a tool whose goal will be to support the task of performing a CW.

## **Résumé**

Prévenir les problèmes d'utilisabilité est toujours une fonction clé dans le processus de conception de logiciels. Nombre de méthodes permettent d'évaluer une interface. Les méthodes réalisant l'examen de l'utilisabilité d'une interface sont des méthodes qui peuvent être accomplies sans la présence d'utilisateurs.

L'objet de notre travail est de présenter en détails une de ces méthodes : la Cognitive Walkthrough. Le premier point que nous aborderons est une explication en profondeur de cette méthode ainsi que sa justification théorique. Ensuite, nous la replacerons dans le cycle de vie du processus de conception d'interfaces. Enfin, nous spécifierons un outil logiciel permettant la réalisation de la Cognitive Walkthrough.

*Before getting to the heart of our work, we'd like to sincerely thank Gilbert Cockton for his cordial welcome in the Informatic Centre of the University of Sunderland (UK) . We thank him for his supervision and his inexhaustible cheerfulness.*

*Then, we thank our thesis supervisor, Mr Bodart, for his precious advices and his patience.*

*We thank our parents for their support and their encouragements.*

*Thanks also to Alice, Christelle, Stephane, Karl, Pat, Eric, Vivien and Marc-Matthieu from Ottignies for the good atmosphere of work they have brought during what could have been our vacations.*

## TABLE OF CONTENTS

<b>Introduction</b>	<b>7</b>
---------------------	----------

---

### *PART 1*

<b>Chapter 1 - A REVIEW OF THE MAIN PROBLEMATIC OF HCI AND REVIEW OF THE D.A. NORMAN MODEL</b>	<b>13</b>
--	-----------

---

1. INTRODUCTION.....	13
2. OBJECTIVE OF THE INTERFACE : UTILITY AND USABILITY .....	13
2.1. EXPLANATION OF THE UTILITY OF AN INTERFACE.....	13
2.2. EXPLANATION OF THE USABILITY OF AN INTERFACE.....	14
2.3. REVIEW OF THE D.A. NORMAN MODEL.....	18
3. CONCLUSION .....	21

<b>Chapter 2 - EVALUATION OF INTERFACES : STATE OF THE ART</b>	<b>23</b>
--	-----------

---

1. INTRODUCTION.....	23
2. THE INTERFACE EVALUATION TECHNIQUES.....	23
2.1. EVALUATION WITH USERS (EMPIRICAL TESTING).....	24
2.1.1. User observation.....	24
2.1.2. The Thinking Aloud Method .....	25
2.1.3. Questionnaires.....	26
2.2. EVALUATION WITHOUT USERS.....	27
2.2.1. GOMS .....	28
2.2.2. KLM (Keystroke-Level Model).....	29
2.2.3. Heuristic Evaluation.....	31
2.2.4. Guidelines Checklists.....	34
2.2.5. Cognitive Jogthrough and pluralistic walktrough.....	35
3. CLASSIFICATION OF THE EVALUATION TECHNIQUES .....	35
4. CONCLUSION .....	36

### *PART 2*

<b>Chapter 3 - THE COGNITIVE WALKTHROUGH METHOD</b>	<b>39</b>
---	-----------

---

1. INTRODUCTION.....	39
2. DESCRIPTION OF THE COGNITIVE WALKTROUGH INPUTS .....	40
2.1. MOCK-UP, PROTOTYPE OR INTERFACE .....	40
2.2. TASK TO ANALYSE .....	41
2.3. THE ABSTRACT TASK .....	45
2.4. AN ANALYSIS OF THE CONTEXT OF USE OF THE INTERFACE.....	45

2.5.	UTILITY AND USABILITY CRITERIA.....	46
2.6.	THE CONCRETE TASK (THE IMPLEMENTED TASK).....	46
<b>3.</b>	<b>FOUNDATIONS OF THE COGNITIVE WALKTHROUGH.....</b>	<b>54</b>
3.1.	INTRODUCTION.....	54
3.2.	STRUCTURE OF THE PRESENTATION.....	54
3.2.1.	The explanation of the question.....	54
3.2.2.	The usual answers to this question.....	59
3.2.3.	The position of the question in the Norman model.....	60
3.3.	SUBGOAL-RELATED QUESTIONS.....	60
3.3.1.	Introduction to the subgoal-related questions.....	60
3.3.2.	Content of the subgoal-related questions.....	63
3.3.2.1.	Question 1 : Will the user try to achieve the right subgoal ?.....	63
3.3.2.2.	Question 2 : What knowledge is needed to achieve the right subgoal ? Will the user have this knowledge ?.....	68
3.4.	ACTION-RELATED QUESTIONS.....	73
3.4.1.	Introduction to the action-related questions.....	73
3.4.2.	The action-related questions.....	73
3.4.2.1.	Question 3 : Can the user perceive that the correct action is available ?.....	73
3.4.2.2.	Question 4 : Will the user associate the correct action with the subgoal they are trying to achieve ?.....	75
3.4.2.3.	Question 5 : Will the user perceive the feedback ?.....	78
3.4.2.4.	Question 6 : Will the user understand the feedback ?.....	81
3.4.2.5.	Question 7 : Will the user see that progress is being made towards solution in relation to their main goal and the current subgoals ?.....	83
3.5.	CONCLUSION ON THE FOUNDATIONS OF THE COGNITIVE WALKTHROUGH.....	84
<b>4.</b>	<b>CONCLUSION.....</b>	<b>85</b>

**Chapter 4 - LOCATION OF THE COGNITIVE WALKTHROUGH METHOD IN THE TASK ANALYSIS 87**

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>87</b>
<b>2.</b>	<b>REVIEW OF THE TASK ANALYSIS.....</b>	<b>87</b>
2.2.	THE INPUTS OF THE TASK ANALYSIS.....	88
2.3.	THE RESULTS OF THE TASK ANALYSIS.....	89
<b>2.</b>	<b>LOCATION OF THE COGNITIVE WALKTHROUGH IN THE TASK ANALYSIS.....</b>	<b>99</b>
<b>3.</b>	<b>COGNITIVE WALKTHROUGH AND THE DESIGN CRITERIA.....</b>	<b>101</b>
3.1.	INTRODUCTION.....	101
3.2.	THE DESIGN CRITERIA AND THE COVERAGE BY COGNITIVE WALKTHROUGH.....	102
3.2.1.	Compatibility.....	102
3.2.2.	Consistency.....	104
3.2.3.	Work load.....	104
3.2.4.	Adaptability.....	105
3.2.5.	Dialogue control.....	106
3.2.6.	Representativity.....	106
3.2.7.	Guidance.....	107
3.2.8.	Error managment.....	108
3.3.	DESIGN CRITERIA VERSUS COGNITIVE WALKTHROUGH.....	110
3.3.1.	Position of the problem.....	110
3.3.2.	Heuristic Walkthrough.....	111
<b>4.</b>	<b>CONCLUSION.....</b>	<b>113</b>

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>115</b>
<b>2.</b>	<b>RESULTS OF THE METHOD AND THEIR INTERPRETATION .....</b>	<b>115</b>
2.1.	THE PROBLEM NUMBER.....	116
2.2.	THE DESCRIPTION OF THE PROBLEM.....	116
2.3.	THE INTERACTION POINT .....	116
2.4.	LIKELY CONCRETE USER DIFFICULTIES IN CONTEXT .....	117
2.5.	THE ASSUMED CAUSES OF THESE DIFFICULTIES .....	117
2.6.	THE SUCCESS/FAILURE CASE .....	117
2.7.	THE SERIOUSNESS OF THE PROBLEM .....	121
2.7.	CONCRETE INTERACTIVE OBJECT CONCERNED BY THE PROBLEM .....	126
2.8.	A DESIGN SUGGESTION TO SOLVE THE PROBLEM .....	126
<b>3.</b>	<b>CONCLUSION .....</b>	<b>126</b>

**PART 3****Chapter 6 - DESIGN OF A TOOL SUPPORTING THE COGNITIVE WALKTHROUGH VIA THE TASK****ANALYSIS WALKTHROUGH**

<b>1.</b>	<b>INTRODUCTION.....</b>	<b>133</b>
<b>2.</b>	<b>CONTEXT ANALYSIS.....</b>	<b>134</b>
2.1.	USER PROFILE DESCRIPTION.....	134
2.2.	ANALYSIS OF THE WORKSTATION .....	136
2.2.1.	Physical environment.....	136
2.2.2.	Tasks allocation.....	136
2.2.3.	Treatment type.....	136
2.2.4.	Task execution modalities .....	137
2.3.	DESCRIPTIVE PARAMETERS OF THE TASKS .....	137
<b>3.</b>	<b>TASKS STRUCTURE .....</b>	<b>139</b>
3.1.	FIRST TASK : ENTER THE INPUTS OF THE CW .....	139
3.2.	SECOND TASK : PERFORMING THE CW .....	140
3.3.	THIRD TASK : CONSULT THE PROBLEM LIST .....	141
3.4.	FOURTH TASK : USE NEW UAN LIBRARY .....	143
3.5.	BOOTSTRAPPING : APPLYING CW ON THE PROJECTED TASK.....	144
3.6.	SEQUENCE OF THE DIFFERENT TASKS.....	145
<b>4.</b>	<b>GUIDANCE .....</b>	<b>146</b>
4.1.	GUIDANCE OF CONTEXT INPUTS.....	147
4.2.	GUIDANCE OF THE TREE CONSTRUCTION.....	148
4.3.	GUIDANCE IN ANSWERING THE CW QUESTIONS .....	150
4.4.	GENERAL GUIDANCE OF NAVIGATION .....	152
4.5.	HELP WHILE PERFORMING THE METHOD .....	152
<b>5.</b>	<b>DERIVATION OF THE ERGONOMICAL SPECIFICATIONS.....</b>	<b>153</b>
5.1.	CHOICE OF THE DIALOGUE ATTRIBUTES .....	153
5.2.	UTILITY AND USABILITY CRITERIA.....	155
5.3.	DERIVATION OF INTERACTION STYLES .....	156
<b>6.</b>	<b>DERIVATION OF THE FUNCTIONAL SPECIFICATIONS :</b>	
	<b>CHAINING GRAPH OF THE FUNCTIONS .....</b>	<b>158</b>
6.1.	TASK 1 : ENTER THE INPUTS OF THE CW .....	158
6.2.	TASK 2 : PERFORMING THE CW.....	159

6.3.	TASK 3 : CONSULT THE PROBLEMS LIST .....	160
6.4.	TASK 4 : USE UAN LIBRARY .....	161
<b>7.</b>	<b>DEFINITION OF THE PRESENTATION .....</b>	<b>161</b>
7.1.	IDENTIFICATION OF THE PRESENTATION UNITS .....	161
7.1.1.	Task 1 : Enter the inputs of the CW .....	161
7.1.2.	Task 2 : Performing the CW .....	162
7.1.3.	Task 3 : Consult the problems list .....	163
7.1.4.	Task 4 : Use UAN library .....	164
7.2.	IDENTIFICATION OF THE WINDOWS .....	164
7.2.1.	Task 1 : Enter the inputs of the CW .....	165
7.2.2.	Task 2 : Performing the CW .....	171
7.2.3.	Task 3 : Consult the problems list .....	174
7.2.4.	Task 4 : Use UAN Library .....	176
<b>8.</b>	<b>BOOTSTRAPPING : APPLYING CW ON THE IMPLEMENTED TASK .....</b>	<b>178</b>
<b>9.</b>	<b>CONCLUSION .....</b>	<b>180</b>

<b>Conclusion</b>	<b>181</b>
-------------------	------------

---

<b>Bibliography</b>	<b>183</b>
---------------------	------------

---

**Appendixes**



## TABLE OF FIGURES

Figure 1 - Schema of the Norman model .....	18
Figure 2 - Proportion of problems found according to the number of evaluators .	33
Figure 3 - Classification of the evaluation technique.....	36
Figure 4 - The three different levels of the task.....	42
Figure 5 - Description of an atomic subgoal .....	48
Figure 6 - ERA-model of the task description used in CW .....	49
Figure 7 - Example of a decomposition of a task into subgoals.....	51
Figure 8 - The result of the task "Drawing a matrix of rectangle" in Microsoft PowerPoint97.....	53
Figure 9 - Concrete goal structure of the early cash dispensers.....	61
Figure 10 - Concrete goal structure of the present cash dispensers.....	62
Figure 11 - Example of the answering questions sequence.....	63
Figure 12 - Exemple of a task selection with Microsoft Word97.....	71
Figure 13 - Goals/subgoals decomposition of the subtask : selecting the lower part of the diagram.....	72
Figure 14 - The hint text of an icon .....	76
Figure 15 - The state of the web browser before clicking in the Email text field ....	80
Figure 16 - The state of the system after clicking in the Email text field .....	81
Figure 17 - The inputs of the task analysis.....	88
Figure 18 - The results of a task analysis, following Pr.Bodart's approach.....	89
Figure 19 - Task description of the abstract task « Drawing a house » on a sheet of paper (abstract model of the task perceived by the user) .....	93
Figure 20 - Task description of the projected task « Drawing a house » in a drawing editor (concrete model of the interface).....	98
Figure 21 - Matrix of the seriousness of the problem according to the utility and usability criteria .....	124
Figure 22 - Interface design evolution with Cognitive Walkthrough.....	127
Figure 23 - Cognitive Walkthrough in the design process.....	129
Figure 24 - Goal/Subgoals decomposition of the task "Enter the inputs of the CW" .....	140
Figure 25 - Goal/Subgoals decomposition of "Performing the CW" .....	142
Figure 26 - Goal/subgoals decomposition of "Use UAN Library" .....	143
Figure 27 - Sequence of the different tasks.....	146
Figure 28 - The data related to the guidance of context inputs.....	148
Figure 29 - The data related to the goal/subgoals decomposition .....	150
Figure 30 - The data related to the answering of questions .....	151
Figure 31 - The help wizard of Microsoft.....	153
Figure 32 - Chaining graph of the functions : Enter the inputs of the CW.....	158
Figure 33 - Chaining graph of the functions : Performing the CW.....	159

Figure 34 - Chaining graph of the functions : Consult the problem list.....	160
Figure 35 - Chaining graph of the functions : Use UAN Library.....	161
Figure 36 - Presentation Units : Enter the inputs of the CW .....	162
Figure 37 - Presentation Units : Performing the CW .....	163
Figure 38 - Presentation Units : Consult the problems list.....	163
Figure 39 - Presentation Units : Use UAN Library .....	164
Figure 40 - Definition of the windows : Enter the inputs of the CW .....	165
Figure 41 - Task 1 : Window11.....	166
Figure 42 - Task 1 : Window12.....	166
Figure 43 - Task 1 : Window21.....	167
Figure 44 - Task 1 : Window22.....	167
Figure 45 - Task 1 : Window23.....	168
Figure 46 - Task 1 : Window31.....	168
Figure 47 - Task 1 : Window32.....	169
Figure 48 - Task 1 : Window33 .....	170
Figure 49 - Task 1 : Extra Window for a guidance service .....	170
Figure 50 - Extra window inserted for a guidance service.....	171
Figure 51 - Definition of windows : Performing the CW .....	172
Figure 52 - Task 2 : Window1.....	172
Figure 53 - Task 2 : Window2 .....	173
Figure 54 - Task 2 : Performing the CW .....	174
Figure 55 - Definition of windows : Consult the problems list.....	174
Figure 56 - Task 3 : Window1 .....	175
Figure 57 - Definition of the windows : Use UAN Library .....	176
Figure 58 - Task 4 : Window41 .....	177
Figure 59 - Task 4 : Window42 .....	177

## *Introduction*

A human-computer interface is an informatic system used by a person to realize a task performed thanks to a set of informatic means through the medium of actions exerted on interactive objects [BODART 98]. But on the interface depends the efficiency with which users will perform the tasks. So, it soon appeared in human-computer interface history that interfaces have to be evaluated to find out problems in their design obstructing a good utilisation.

Many ways, in many forms, are used to evaluate interfaces. This aim of this work is to present one usability inspection method that permits to discover these problems. This method is the Cognitive Walkthrough.

So, after synthesising the problematic of HCI, the objectives of this work was to study first what was made by authors on this method. After being used to it, we have developed the first main point of this thesis, **the theoretical foundation of the Cognitive Walkthrough method**. Then, we began to discover its flaws and weaknesses. As often as possible, we try to give solution to this problem. The second objective of our work is **to try to place the Cognitive Walkthrough in a global design process**.

The result of our work is the presentation of the method with all the guidance needed to understand it, and the theory behind it, to perform it in the most efficient way and help to make it the most productive as possible. This will be a critical presentation of Cognitive Walkthrough, i.e. critics, limits of the method and our contribution to it have been included from beginning to end of this thesis.

## *Organization of the thesis*

This thesis is organized in three different parts. The **first part** concerns the field of the evaluation. It contains a review of the main problematic of Human-Computer Interaction and a state of the art in the domain of evaluation of interfaces.

- We will start this part by recalling the objectives that a good interface must reach. In this *first chapter*, we will also present the task model of Norman that will justify pertinence of the Cognitive Walkthrough.
- *Chapter 2* presents the most used techniques to perform an evaluation of an interface and to find out usability problems. We will also show a classification of these methods.

The **second part** is the presentation of one specific usability inspection method, called Cognitive Walkthrough.

- The Cognitive Walkthrough itself is the subject of the *third chapter*. The different input components are explained as the way to perform the method. The objective of this chapter is also to justify the method according the Norman model of the task.
- *Chapter 4* aims to locate the Cognitive Walkthrough in the task analysis and in which way this evaluation method takes into account the design criteria.
- Finally, *Chapter 5* finishes this second part by defining the outputs of the method. This chapter will also bring a final conclusion about the possible location of the Cognitive Walkthrough in the task analysis.

Finally, the **third part** aims to give the specifications of an editor whose goal is to support the task of performing one task of evaluation of an interface thanks to

the Cognitive Walkthrough. The analysis of this task is presented with the recommendations related to the needed guidances of this tool.

# PART 1

## *Chapter 1*

### A REVIEW OF THE MAIN PROBLEMATIC OF HCI AND REVIEW OF THE D.A. NORMAN MODEL

## **1. Introduction**

The aim of an interface is that the user achieves his task efficiently, i.e. that the interface must be useful and usable. These two notions will be explained in the next point of this chapter.

A task is an activity, realized by a person or a group of persons working together, possibly thanks to computers and other equipments, in order to bring (in a given field) about a change of state, corresponding to a main goal to reach ([BODART 98]).

## **2. Objective of the interface : Utility and usability**

### **2.1. Explanation of the utility of an interface**

An interface is useful if it provides users with the necessary functionalities to perform in the best way the tasks assigned to them. Users want to find in the software (via the interface) the functionalities that correspond to the functionalities of the real task if this one exists. Thus, the notion of utility is located in the side of the user. Indeed, it's throughout the interface that the user accesses the application and the system ([BODART 98]).

### *Example*

*In the early time of software of e-mail, users can not easily attach a document whereas in the real task of sending mail via the postal services, users can send documents coming with the mail.*

## **2.2. Explanation of the usability of an interface**

The usability is the efficiency, fiability and satisfaction with which specific users reach specific goals in a specific environment [ISO 92]. An interface will be usable if it is compatible with the cognitive profile of the user. Thus the interface doesn't restrict user actions to something strange given the nature of the task ([BODART 98]).

### *Example*

*With the Software Eudora, when users receive an e-mail with a document (file) in attach, they have to note the name of the document and after that, they have to look for in « Eudora/attach/name\_of\_the\_file ». This is an action completely strange to the nature of the task.*

*In the other side, with Eudora Pro, if users receive an e-mail with an attach file, this one is represented by an icon and users can open it directly by clicking on it. This is closer to the real task of receiving mail with an attached document.*

So, we can say that it is very important to assess an interface with regard to its utility and usability.

These two criteria put the main problematic linked to the creation of an interface in evidence. Indeed, there are two « worlds », two main levels of task :



## 1. The abstract task :

The abstract task concerns the structure thought by users. It is the structure that represents the way for users to accomplish the task.

In other words, it's the mental representation (mental model) of the task perceived by users. If the task has a real origin, e.g. to send an e-mail has the origin of sending a mail via the postal services, the mental representation of the task is the way performing the task in real life (e.g. how to send a mail via the postal services ?).

If the task hasn't a real origin, e.g. to increase the size of the virtual memory in Windows, the mental representation of the task is the way that the user will think about performing this task, but not in interface terms.

### *Example*

*We consider an interface of a GPS which is implanted in a car to help drivers to find an itinerary. This task has a real origin. The main points of the abstract task are :*

- 1. To select the departure and the arrival.*
- 2. To select the criteria of time or rapidity.*
- 3. To select if the user wants a free itinerary or not.*
- 4. To keep in mind the weight and height of the car permitting to select the taken routes.*

## 2. The concrete task :

The concrete task is the goal structure of the task as the system requires to complete it to achieve the task. In other words, it's the goal structure of the task imposed by the interface.

When we will speak about this level of task, we will call it « the concrete task » or « the implemented task ».

Finally, let's notice that these two structures are goal structures which are likely different.

The aim of an interface is to reduce the distance between these two worlds. The concrete task, fixed by the interface, has to match as good as possible to the abstract one. So, the interface has to fit as much as possible to the user and not the opposite.

In conclusion, we have already said that the quality of an interface can be inferred from its useful and usable aspect. We have to our disposal six utility and usability criteria. The first fifth are proposed in [SCHNEIDERMAN 92] and the last one is proposed further by [BODART 98]. They permit to evaluate an interface (i.e. its quality). Here is the walkthrough : we must give a balance to each of the criteria according to the importance and context of use of the task. This balance of each criterion is a direct result of the task analysis<sup>1</sup>.

Here is a presentation of the various criteria :

### **The learning time**

It depends on the frequency of use of the task. The more the frequency is high, the more the learning time may be high. Moreover, if the frequency is high, the learning time can be eventually higher and will reduce when number of uses will grow. If the frequency is low, the learning time must be also little because it would be inadmissible to oblige the user to take too much time learning again the task at each time.

### **The rapidity of execution**

It's the rapidity to perform the task.

### **The error rate performed by the user**

We can consider :

- The frequency of errors. There are two kind of errors :
  - the errors of execution : linked to problems of manipulation (syntactical errors)
  - the errors of intention : the user selects an unappropriate control because of a bad interpretation.
- The time of correction : it's the time to correct an error.

### **The period of persistency**

It's the time during which the user keeps the acquired knowledge.

### **The subjective satisfaction of the user**

The sensation of pleasure and comfort that the user perceives using the interface.

### **The degree of coverage of the mechanisms of the interface**

The degree of coverage is high if the functions provided by the system through the interface cover the most functions desired by the user to perform his task. This notion is directly related to the utility of an interface.

---

<sup>1</sup> See Chapter 4 for more details.

### 2.3. Review of the D.A. Norman model

We have seen that it is important, for the utility and usability of an interface, to reduce the effort required by the user to reduce the distance between the user mental representation of the task (abstract task) and the system's physical representation of this same task, which is now implemented (concrete task). This distance has been specified among others by the D.A. Norman model [NORMAN, DRAPER 86] (Fig. 1). This model really puts these two worlds in evidence and there are these two worlds, and more precisely the gap between these two worlds, which are in the centre of the main problematic of HCI. Thus, it is obvious that every assessment method centered on the task of an interface must fit this model in the purpose to « measure » the importance of this distance between these two worlds. The more this distance will be short, the more the interface will be of high quality.

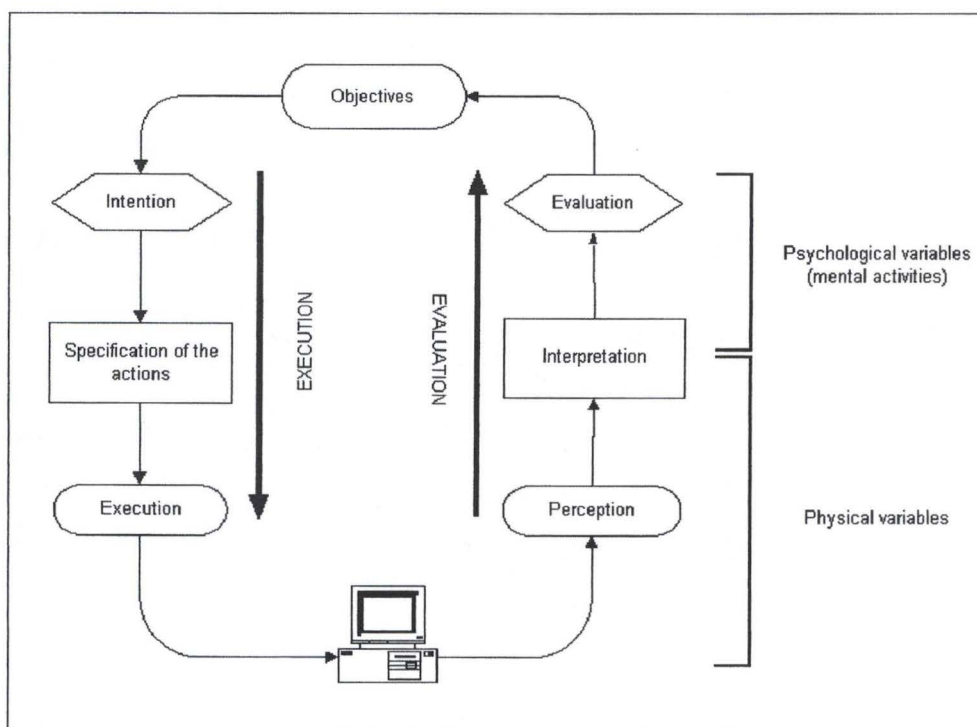


Figure 1 - Schema of the Norman model

Let's notice that this distance includes two notions : the semantic distance and the articulatory distance. The semantic distance puts a question to know if we can express in the language of the interface what we want to achieve. The articulatory distance puts a question to know if we can infer from the form of an expression its meaning. These two distances can be assessed in two levels : the execution and the evaluation levels. We don't go further about these two notions in our thesis. See [BODART 98] for more details.

The foundation of the Cognitive Walkthrough is related to a model of the task such as the model of D.A. Norman. This foundation will be explained in point 3 of chapter 3.

As this model is important to understand how the steps of the method we will present have been chosen, we will now explain the model in details. The model consists in seven iterative steps :

1. To set the objective
2. To define the intentions associated to the objective to reach
3. To specify the sequence of actions
  - a) To translate the objectives and intentions to desired states of the system
  - b) To find out the devices of the mechanics of the interface which will produce this state
  - c) To find out the required manipulations of these mechanics
4. To execute the sequence of actions
5. To perceive the state of the system
6. To interpret the state of the system
7. To assess the state of the system

When users want to achieve a task using a system (an implemented task), they have in mind an objective to achieve. After defining the intention to achieve the objective, they have to express their objective in term of a desired state of the

system, to determine the means to reach this state and how to manipulate these means. But there is a gap between the objectives of the user and the sequence of actions to trigger off to achieve this objective. This is the execution gap. To reduce this gap, the interface should provide objects, commands, prompts and means as close as possible to the mental representation of users (the abstract task). So, the cognitive effort made by users to translate their objectives into sequence of actions in the interface would be shorten.

After executing the sequence of actions, the system provides feedback which is the result of the execution. This feedback is provided by physical variables (for instance, visual output). Users have to perceive this new state of the system and to interpret the physical state into mental state (i.e. an highlighted icon means that the tool provided by this icon is selected). Then, they have to evaluate the result obtained according to the primary objective. This is the evaluation gap. The interface should provide output means whose conceptual model is easily perceived, interpreted and evaluated by users : the smaller the gap, the smaller the cognitive effort required to preceive and evaluate the feedback in relation to the objective.

This model is iterative which permit to take in consideration a decomposition in goal/subgoals as we need in the Cognitive Walkthrough.

When we will present all the steps of the accomplishment of the Cognitive Walkthrough in chapter 3, we will replace each one according to the Norman model.

### **3. Conclusion**

We have seen the main problematic of the HCI in this chapter and the model [NORMAN, DRAPER 86] which provides a framework for this analysis.

An interface almost always includes errors, lacks of guidance and so on. These errors diminish the interface usability. So it's very important for analysts and designers to have methods which permit to assess an interface to find out the mistakes of it and repair it.

## EVALUATION OF INTERFACES : STATE OF THE ART

### 1. Introduction

The evaluation of an interface is an assessment of the utility and usability of this interface. An evaluation permits to find out utility and usability problems that might prevent users from achieving their tasks.

*« Usability problems are aspects of the system which could reduce the usability of the system for the user, for example to confuse them, to slow them down or stop them completing their tasks » [LAVERY, COCKTON 97].*

Many techniques can be used to perform this evaluation. The differences between these techniques turn on their focus, the person responsible for the evaluation,...

In this chapter, we will present the most famous and used usability techniques that are used for interface evaluation today. A classification will also be proposed to give a global view of the differences between the evaluation methods and to locate the Cognitive Walkthrough among them.

### 2. The interface evaluation techniques

Lewis and Rieman in [LEWIS 93] classify the evaluation techniques in two groups : the techniques with users and the ones without user.



## 2.1. Evaluation with users (empirical testing)

As the system being designed must permit users to realize their task, the implication of users in the design process is unquestionable.

*« No matter how much analysis has been done in designing an interface, experience has shown that there will be problems that only appear when the design is tested with users. The testing should be done with people whose background knowledge and expectations approximate those of the system's real users. » ([LEWIS 93]).*

The profile of the test users are to be close to those of users who will actually use the designed system. For instance, architecture software whose goal is to support the drawing of building plan should be tested by architects themselves, who are the final users.

User testing is probably the best method to find out usability problems caused by an interface. An user is placed in front of the interface and is asked to try and perform one or more tasks that the system intends to support. Users who will participate to the test must be carefully choosen.

Three kinds of user testing can be differentiated ([FARENC 97]): user observation, the think-aloud method and the questionnaires.

### 2.1.1. User observation

The test, performed in a real situation of use or in a usability lab, is directed by an expert who takes notice of the usability problems encountered by the test user. Data are collected by recording on the fly with the usability expert writing down his remarks, recording for instance on videotapes, or think-aloud techniques. To be optimal, such techniques require the involment of all the intervening parties, i.e. users, usability evaluators and system designers.

## Advantages

User observation, done with a sufficient number of users, usually brings good results in terms of usability problems found.

## Disadvantages

A first drawback of this technique is the time required to perform not only all the necessary user testings to find out a fair number of problems but also the time needed to do one single test. A videotape-recorded one-hour test often leads to an analysis by the usability expert lasting about ten hours. For one task performed by one user ! And a time-consuming method means an expensive one.

### 2.1.2. The Thinking Aloud Method

The principle of this technique is simple.

*« You ask your users to perform a test task, but you also ask them to talk to you while they work on it. Ask them to tell you what they are thinking: what they are trying to do, questions that arise as they work, things they read. » [LEWIS93].*

During the course of a test, where the participant is performing a task or scenarios (a set of specific tasks that can be accomplished by mean of the interface), the participant is asked to vocalize his thoughts, feelings, and opinions while interacting with the product system [WOODWARD 98].

As User Observation, the Thinking Aloud Method collects data allowing to evaluate the interface. During the test, the evaluator should note information about what users have found confusing, their actions (and the reasons of those actions) and consequently the usability problems with sometimes their cause(s).

During the test, the evaluator must not keep a passive attitude. He must force users to give a good flow of information by asking questions such as « What are

you thinking now ? », « Why did you choose this button ? »,... Once again, the test can be video recorded, transcribed into written notes or both.

### **Advantages**

This approach is widely used in computer design as it brings valuable data and locates where users have difficulties in the interface. Indeed, as evaluators see users while interacting with the system, they can notice not only areas of confusion and unforeseen events but also why these occurred. Direct aloud feedback gives specific information about the interface.

Further, the position of the think-aloud technique in the lifecycle, from mock-up to finished interface, suits all the design process.

A last advantage of this evaluation technique is located in the expert's required background : this one is low as think-aloud can be used with little training or usability experience, as long as leading questions and other forms of 'priming' are avoided.

### **Disadvantages**

The analysis of verbal reports is difficult as results depends on the interpretation of users' remark and on the faculty of the user to verbalize his thoughts. The fact that users' answers can be influenced by the way of asking them questions is a second drawback of this method [FARENC 97].

#### **2.1.3. Questionnaires**

Questionnaires are written lists of questions distributed to users in order to answer them after testing. The aim of questionnaires is to collect data about users' impressions after they used the system. This is thus an a posteriori method.

As soon as after users tested the system, they are asked to answer questions to determine the level of their satisfaction.

### **Avantages**

This method is rather cheap and can be easily used at a wide scale : once the questionnaire has been written, it can be joined to every use of the system.

### **Disadvantages**

Whereas this method permits easily to collect information in an unexpensive way, the difficulty lies in the redaction of good questions and in the interpretation of the answers [FARENC 97].

This method is often used to get feedback from final users after they acquired the system and so, to correct problems in future versions. In this point of view, the design is already complete when problems are revealed [HOM 98].

## **2.2. Evaluation without users**

Evaluation with users, even if it brings good results, is not always possible. Three reasons can be distinguished. Firstly,

*« Users' time is almost never a free or unlimited resource. Most users have their own work to do, and they're able to devote only limited time to your project. When users do take time to look at your design, it should be as free as possible of problems. This is a courtesy to the users, who shouldn't have to waste time on trivial bugs that you could have caught earlier. » [LEWIS 93].*

Secondly, user testing is actually efficient when the test is performed by a sufficient number of users. As each user finds a subset of all the problems, those are likely to be uncovered if only a small number of users is available.

Finally, as said earlier, user testing is an expensive method in terms of as much number of user tests as time required to analyze the result of one test.

There are many user-free evaluation methods (usability inspection methods). A distinction can be made between the task-centered methods and the task-free methods. The first class contains methods such as GOMS, KLM, Cognitive Walkthrough and the methods derived from it. The second one includes Heuristic Evaluation and the Guidelines evaluation methods. This enumeration is not exhaustive ; we limited ourselves to the best known and representative existing methods. Some explanations will be given below for each of the cited methods.

### 2.2.1. GOMS

GOMS is an acronym that stands for GOALS, OPERATORS, METHODS, and SELECTION RULES. Goals represent the goals that an user is trying to accomplish, usually specified in a hierarchical manner. The task is decomposed into goals, subgoals and elementary subgoals. Operators are the set of atomic-level operations with which an user composes a solution to a goal. Methods represent sequences of operators, grouped together to accomplish an elementary goal. Selection Rules « if...then » are used to decide which method to use for solving a goal when several are applicable [WOOD 98].

GOMS predicts task times for an expert user of the system. Task times are obtained by adding up realization time of each physical and mental step required to achieve the task.

In fact, GOMS is a family of techniques that

*« can provide much insight into a system's usability, such as, task execution time, task learning time, operator sequencing, functional coverage, functional consistency, and aspects of error tolerance »  
[WOOD 98].*

### **Advantages [JOHN, KIERAS 94].**

GOMS models are usefully approximate, make *a priori* predictions : they don't need any working interface or mock-up. It is interesting as these predictions will be used to develop the specifications of the system to design.

Second, the complexity and efficiency of the interface procedures is addressed very well by models in the GOMS family.

Third, GOMS models are especially useful because one way to characterize a whole task is to describe the procedures entailed by the whole task; this allows the individual aspects of the interface to be considered in the entire task context.

Finally, GOMS has been proven to be learnable and usable for computer system designers .

### **Disadvantages**

Firstly, GOMS only considers predictions about error-free task times performed by expert users. It does not apply to beginners.

Performing a GOMS technique for a task of some importance may require a daunting load of work. Thus, covering the whole interface is difficult.

#### **2.2.2. KLM (Keystroke-Level Model)**

Keystroke-Level Model is an ancestor of the GOMS techniques. As being a task-centered inspection method, KLM forces the evaluator to take a close look at the sequence of actions performed by users. The aim is to compute the time required for an experienced user to perform a given task. To predict task times, the evaluator totals the time required to realize each physical step composing the

task, each keystroke. Unlike other GOMS techniques, mental steps are not taken into consideration. KLM is thus a simplified method.

The evaluator must access data about the predictions of times for each step and keystroke. These data are constituted by measuring the average performing time for each physical step. This average is calculated from the observation of many users while performing these steps.

*Example*

<b>Physical steps</b>	<b>Time required (in seconds)</b>
Enter one keystroke on a standard keyboard	0.28 second
Use mouse to point at object on screen	1.5 second
Move hand to pointing device or function key	0.3 second

Excessively high task times, comparing to similar complexity task, may reveal a problem. Furthermore, this method highlights different problems according to the task description and the sequence of actions chosen by evaluators.

KLM is especially recommended for widely used procedures. A small time saving for a procedure performed thousands of times by a large number of users can be favourable [LEWIS 93].

**Advantages**

This evaluation method can easily be used by a non-expert evaluator. By only considering physical steps of a task, KLM requires no cognitive knowledge and can be performed quite automatically.

## Disadvantages

However, KLM is burdensome as a task of several minutes is to be translated into hundreds of physical steps. A lot of time will be thus spent on detailing the task and then on adding up the times of these numerous steps.

Further, KLM only considers the physical steps composing tasks. Mental steps (perception, choose to achieve a subtask,...) are not taken into account.

### 2.2.3. Heuristic Evaluation

Heuristic Evaluation is an usability inspection method where features of the interfaces are examined as corresponding to a list of heuristics. In opposition to guidelines which usually include thousands of rules, HE contains a small list of basic principles. The problems found by this technique correspond to violations of the heuristics. [Nielsen 94] defines the following list of heuristic principles :

- *Visibility of system status*  
The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- *Match between system and the real world*  
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- *User control and freedom*  
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.



- *Consistency and standards*  
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- *Error prevention*  
Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
- *Recognition rather than recall*  
Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- *Flexibility and efficiency of use*  
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- *An esthetic and minimalist design*  
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
- *Help users recognize, diagnose, and recover from errors*  
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
- *Help and documentation*  
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information

should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

HE is one of the most used evaluation methods.

### Advantages

Only a few training and time from the evaluator is necessary. Furthermore, it usually brings good results, i.e. finds many usability problems.

The small list of heuristics to apply lets the method be applied quickly, whereas a long number of rules would make it time-consuming and expensive.

In principle, HE can be performed by one single evaluator. But as like many of the usability inspection methods, many more usability problems are found when the method is made separately by some different evaluators and when the results are then aggregated. The following figure [NIELSEN 93] (Fig.2) shows the proportion of problems found according to the number of evaluators whose HE results are added up.

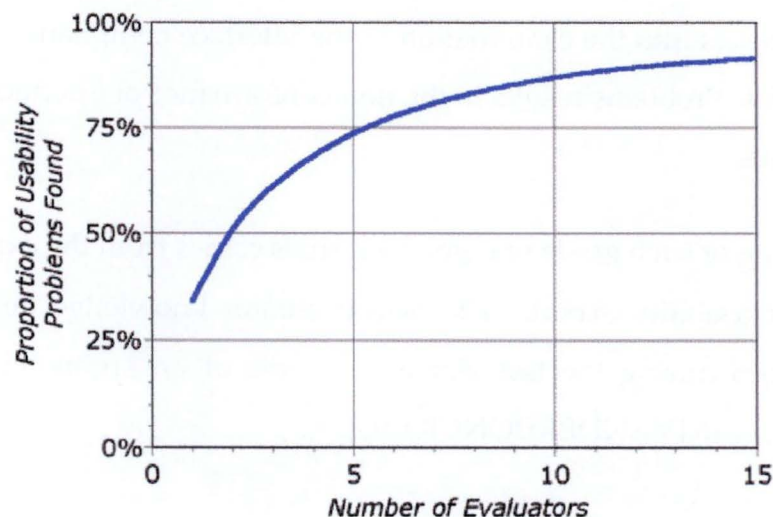


Figure 2 - Proportion of problems found according to the number of evaluators

So, a high number of evaluators will find more problems but the evaluation will prove to be expensive.

Furthermore, as evaluator are not to perform a task to evaluate the system, HE can be used early in the design process. As early as a first mock-up is designed, the method can be applied.

### **Disadvantages**

The HE method doesn't provide much guidance about how to conduct the evaluation, which features look at first. Evaluators have to decide on their own how they want to proceed with the evaluation. It is almost the opposite of task-centered methods that force evaluators to focus on specific parts of the interface, HE mainly finds general, recurrent problems but not the smaller and the more particular ones.

#### **2.2.4. Guidelines Checklists**

Guidelines Checklists usually brings together thousands of ergonomic rules. The evaluation consists into the examination of the interface compliance to the list of ergonomic rules. Problems results in the non-conformance of interface features to ergonomic rules.

The constituting of such guide of ergonomic rules comes from the experience and knowledge of usability experts. The own evaluator knowledge helps to apply these guidelines during the test. For an example of an ergonomic guide, the reader may refer to [VANDERDONCKT 93].

## **Advantages**

Like Heuristic Evaluation, this method positions itself early in the lifecycle. As soon as a mock-up is available, it can be evaluated (no task is required).

## **Disadvantages**

Because of the importance of the evaluator experience to use the method, Guidelines are not really accessible to novice evaluator. Moreover, the scanning through thousands of ergonomic rules for each interface part and features is not an easy and quick way to find usability problems.

### **2.2.5. Cognitive Jogthrough and pluralistic walkthrough**

These two methods are derived from the method we will study in details in this thesis : the Cognitive Walkthrough method.

These two methods are task-centered and required either expert evaluators or non-expert ones. These methods are more detailed in [LEWIS 97].

## **3. Classification of the evaluation techniques**

The following table classifies the different evaluation techniques we described in the previous section. As said before, the main distinction between the techniques concerns the involvement (or non-involvement) of users in the evaluation process. In each of these two categories, we distinguished the task-centered techniques and the task-free ones.

We made another distinction in relation with the quality of evaluators required : expert and non-expert. Then we took into account the number of evaluators who are necessary to perform the techniques.

Quality of the evaluator		WITH USER		WITHOUT USER	
		Task-centered	Task-free	Task-centered	Task-free
EXPERT	1	Questionnaires	Opinion poll without task scenario	GOMS, Cognitive Walkthrough, Cognitive Jogthrough <sup>1</sup>	Guidelines
	N	User Observation <sup>2</sup>		Pluralistic walkthrough <sup>1</sup>	
NON-EXPERT	1	Think-Aloud		KLM	Heuristic evaluation
	N	User Observation <sup>1</sup>		Cognitive Jogthrough <sup>1</sup> , Pluralistic walkthrough <sup>1</sup>	

Figure 3 - Classification of the evaluation technique

Cognitive Walkthrough is the subject of this work. That's why we didn't explain it in the previous section. We only say here that this is a task-centered evaluation technique to be performed by one expert usability evaluator and which lets him see how an interface would appear to a first-time user, i.e. an user who hasn't never faced this interface. All this will be more explained in the next chapters.

## 4. Conclusion

There exists a lot of assessment methods. Each method has its own range, advantages and disadvantages. We will present the method we have learned, the Cognitive Walkthrough, in the next chapter.

<sup>2</sup> These techniques are group evaluation techniques that require both expert and non-expert participants.

# PART 2

THE COGNITIVE WALKTHROUGH METHOD

## **1. Introduction**

We will now present the Cognitive Walkthrough method. The original Cognitive Walkthrough was developed by Clayton Lewis, Peter Polson, John Rieman and Cathleen Wharton [WHARTON et al. 94]. The Cognitive Walkthrough method we present here has been extended by Darryn Lavery and Gilbert Cockton [LAVERY, COCKTON 97]. We brought also some contributions to it, which will be presented as long with the method.

This chapter will be divided into two main parts.

First, this chapter is aimed to give the required inputs to perform a Cognitive Walkthrough. Secondly, its purpose is to explain how to perform a Cognitive Walkthrough method : the analyst is guided in a strong manner through the different steps of the method. The term « walkthrough » takes here all its sense. Another goal of this third chapter is to provide a theoretical justification of the method by referring to the Norman model of the task presented in Chapter 1.

An example of an application of the Cognitive Walkthrough is available in the Appendix B. This will permit to clarify the use of the method.

## **2. Description of the Cognitive Walkthrough inputs**

What is needed to perform a cognitive walkthrough includes

- A mock-up, a prototype or an already-made interface which gives a good idea of the final interface ;
- A task to analyse ;
- The abstract task;
- An analysis of the context of use of the interface ;
- Prioritised utility and usability criteria (a balance of the utility and design criteria) ;
- The concrete task (the implemented task);

### **2.1. Mock-up, prototype or interface**

The mock-up, prototype or interface is what should be evaluated. The mock-up of the interface may only consists in a series of screens displayed during the task execution. But in this case, some information can be missing, for instance feedback information. So, the analyst, performing the Cognitive Walkthrough method on a mock-up must be conscious of this.

In the case of using a prototype, the prototype of the interface is an incomplete interface where all the screens that intervene in the task execution and the feedback associated with the actions presented in these screens are present. The tool or prototype version which is used to perform the analysis must be specified.



## 2.2. Task to analyse

Cognitive Walkthrough is a task-centered usability inspection method, i.e. it evaluates the usability of an interface by having a close look at specific tasks provided by a system.

This section is about the problem related to the choice of a task on which a Cognitive Walkthrough will be performed.

*“Choosing the right tasks to examine is key, since aspects of the interface not involved in the tasks that are chosen will not be examined” [LEWIS 97].*

Before explaining how to choose the task, it's better to say more about this notion of task.

During a design process, the analyst go through three different levels of task : abstract, projected and implemented (concrete) (Fig. 3). We have already presented the first and last level in the chapter one when we presented the Norman model [NORMAN, DRAPER 86].

The abstract task is the level of task as people perceive it in their environment. For example, an abstract task for a sightseer may be to find a way between two cities using a map and according to certain criteria, i.e. the shortest path, the cheapest path or the path including the biggest amount of sightseeing places.

Supposing that an analyst wants to develop a program supporting the task of finding a way between two cities following some criteria, he has to perform an analysis of users, their environment (the workstation) and, of course, the abstract task. From these information, is derived the projected task. The projected task includes the task structure as it will be implemented in the system being developped.

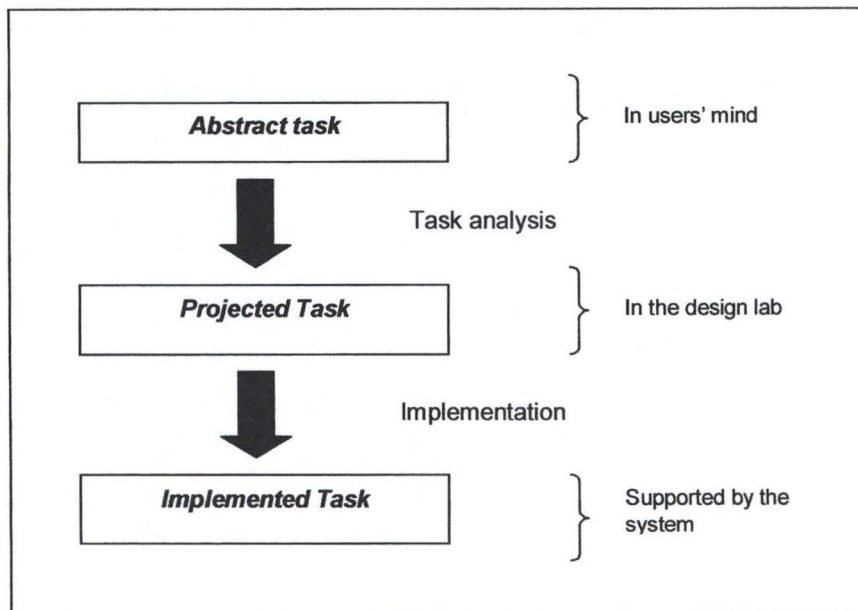


Figure 4 - The three different levels of the task

Following the design walkthrough proposed by Professor Bodart [BODART 98], the ergonomic and functional specifications should be analysed. From the first one, we can describe the interaction style and the dialogue attributes. From the second one, the information and functions specifications and the chaining graph of the functions can be derived. The chaining graph of the function is a multigraph where nodes are functions or messages and arcs are relations of precedence. From all the described specifications, we implement a concrete interface supporting the projected task and thus the abstract one. We are now at the third level of the task : the implemented task. So, the implemented task includes all the user actions on the interface mechanisms to complete it.

We need to select a task to analyse which is intended to be supported by the interface according to the users' requirements : the set of tasks supported is also provided by the task analysis.

To choose tasks on which to perform Cognitive Walkthrough, it is necessary to distinguish two kinds of softwares : generic and specific.

For a **generic software** (e.g. a word processor, a drawing editor,...), where the number of the different tasks available can really be huge, it seems to be impossible to realize an evaluation for all of them : this number of tasks risks becoming too high and make Cognitive Walkthrough a time-consuming and too expensive method [JEFFRIES et al. 91]. Two criteria are useful to choose the tasks needed.

Firstly, the important tasks are to be chosen. You have to choose the central tasks of the system, i.e. those which the system was built around.

*"Important tasks are those that are most frequent or infrequent but critical tasks." [LEWIS 97].*

Secondly, the tasks should be realistic, i.e. the most likely to be done by the user. To do so, it is important that the evaluator gets in touch with users to see how they will accomplish the task because perhaps users will realize quiet different tasks from those envisaged in lab [JOHN, PACKER 95]. Furthermore, the chosen tasks should be realistic, using multiple functions of the system if required. Because functions can be supported separately but a problem may happen when used together. Tasks which don't cover some functions won't show that. Just take the most

*"common tasks that cover the most important uses with realistic complexity".[LEWIS 97]*

For a **specific software** which depends on a context and so could have reduced functionalities (for instance, a GPS program where the different functions are : to find a way between two cities, to modify the way and to find out sightseeing information about places,...), the better is to perform a cognitive walkthrough for all the important provided functionalities (that are derived from the task analysis).

The result of this part is a text description of the chosen task.

### **First remark**

Usually a projected task is implemented in one or more ways. An interface often offers different mechanisms according to the level of user's expertise. For instance, a beginner-user will go through the functionalities using the mouse and the pop-up menus. On the other side, an advanced user would prefer quick shortcuts and keyboard manipulation.

So, having in mind the profile of users, their knowledge of the system,... one can choose between the task alternatives.

### **Second remark**

The projected task should support completely the abstract task : all the functionalities needed to achieve the abstract task must be taken into account in the projected task. By selecting tasks at the implemented level, the role of Cognitive Walkthrough is restricted to an ergonomical evaluation of the interface and so the Cognitive Walkthrough is not central to a design development process but only an 'add-on'.

Indeed, if the above assumption is broken, and thus all the functionalities needed to perform the abstract task are not supported in the projected task (if the developer does not his job perfectly), then we have to choose task to analyse at the first level. By doing so, performing Cognitive Walkthrough shall put in evidence the lack of utility, of mechanisms in the interface to achieve some parts of the abstract task. In this case, the problem is not located in the interface design anymore but in the task analysis.

### **2.3. The abstract task**

It is the decomposition in goal/subgoals for the task in the mental model of the user as we have already defined it in the chapter 1. This information can also be taken from the task analysis. This decomposition is needed because it permits to compare the abstract task with the projected or implemented (concrete) one. This will be assessed in the Cognitive Walkthrough method.

### **2.4. An analysis of the context of use of the interface**

It is important before performing the cognitive walkthrough to have some information about the context of use of the interface.

The information we need about the context of use of the interface are the user profile and the context of work. These two information are, among others, produced by the task analysis [BODART 98]. Having in mind the characteristics of the different users is important while performing the Cognitive Walkthrough [FRANZKE 95].

As explained in [BODART 98], the user profile contains attributes describing the user's experience of the abstract task, the user's experience of informatic systems, the user's level of motivation to use the system and the user's experience of a complex interaction technique. To these attributes, we add the user's experience of the implemented task on a previous or another system.

The context of work includes an evaluation of the physical environment (equipments, environment and work conditions such as visibility, noise, ...), the task allocation and single or multi-task allocation (i.e., will the task be interleaved with and/or interrupted by other work tasks ?).

## **2.5. Utility and usability criteria**

The utility and usability criteria, also coming from the task analysis, are used to interpret the results of the Cognitive Walkthrough method but are not used to perform the method. By « interpret », we mean associate a level of severity with each usability problem found in regards with the utility and usability criteria.

It's important to understand that the utility and usability criteria are independent of the interface, of the implemented task. We are assessing here the importance of the abstract task in relation to the utility and usability criteria we have described in the first chapter, point 2. These criteria are balanced according to the importance of the task and the context of use of this task.

## **2.6. The concrete task (the implemented task)**

This part includes a goal-subgoal decomposition diagram and a formal description of all the actions which compose the lowest level subgoals [LIVERY, COCKTON 97].

A goal-subgoals decomposition starts from the main goal of the user. The main goal of the user is to reach the final state of the system corresponding to the objective for which the task is performed.

Achieving this goal requires the accomplishment of some subgoals. A subgoal is an intermediate state of the system to achieve in the process leading to the main goal. These subgoals can also be decomposed into further subgoals. The decomposition of a subgoal stops when it can't be divided into other subgoals. This final subgoal is only made of a set of actions, we say that they are atomic. This kind of performing a task decomposition is close to the decomposition obtained with the TKS model [JOHNSON 92].

First of all, we will define in the following paragraphs what we mean by "action".

Firstly, an action consists of an operation executed by users via an interaction technique. For instance, a keyboard, a mouse, an optical pen or any other tool permitting users to interact with the interface.

Secondly, the action must have an impact on either the interface (when it triggers off a feedback) or the state of the system. We will explain more about this last point. For example, in a drawing editor, after a mouse click on an icon (i.e. to draw a shape), this icon is selected. Users can thus use the function, related to this icon, which is now selected and ready to be used. But the interface can be made in such a way that no visual feedback is provided to users. Only the state of the system has changed.

Naturally, an action can have both impact on the interface and the state of the system.

For each undecomposed subgoal, a description of the composing actions is to be provided. For each action, the system response and any comment about the system state is to be given. Figure 5 presents such a description.

<b>Method to achieve subgoal X : Name of the subgoal</b>			
Main goal and active subgoals :			
A. Main goal			
{active subgoals}			
...X. Name of the subgoal.			
Comments : any necessary comments. For instance, preconditions.			
	<b>User action</b>	<b>System Feedback</b>	<b>Comments</b>
Action No  (e.g: X1)	Describe here the user action	Describe here the system's feedback to the action performed	Describe here system status, references to screenshots,...

Figure 5 - Description of an atomic subgoal

Note the Action No with the letter (let's notice that this may be a figure) of the subgoal suffixed with a number representing the order in the action sequence.

The user action can be described in prose. But as this is time-consuming and as such a description is sometimes ambiguous, it is preferable to use the User Action Notation [DIX , HARTSON 93]. UAN is a quick, expandable and unambiguous method to describe user action but it takes some time to learn it (see Appendix A for a short description of the UAN language).

Describing the system feedback and state can also be done using UAN. A complete task description is available in Appendix B.

Here follows an ERA-model of the task description that we will use to perform a Cognitive Walkthrough (Fig. 6).



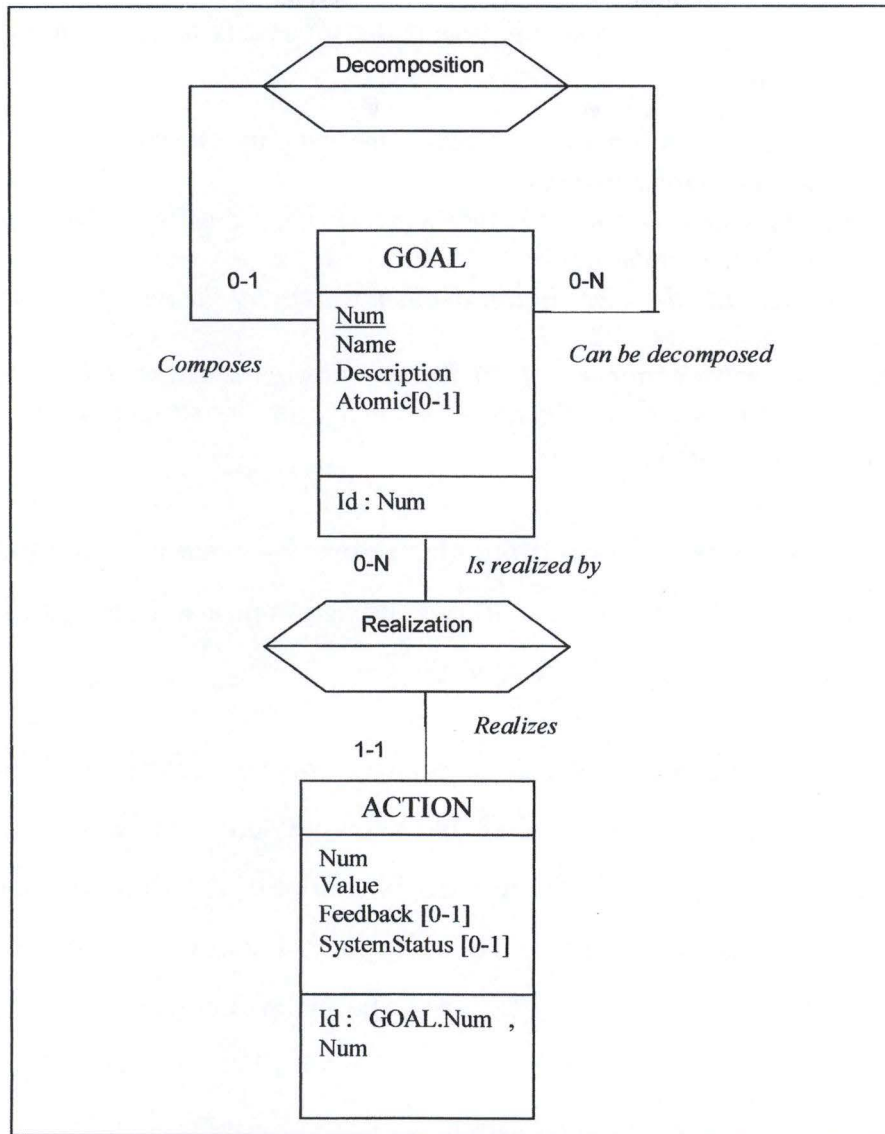


Figure 6 - ERA-model of the task description used in CW

*Integrity constraints :*

*Only one goal (the main one) doesn't compose other subgoals.*

*If a subgoal is atomic then this subgoal can't be decomposed into other subgoals.*

*If a goal is not atomic, it can't be decomposed into actions.*

*If a goal is atomic, it must be decomposed into actions.*

*Explanation of the components of the scheme :*

*GOAL : this entity describes a goal intervening in the goal/subgoal decomposition.*

- *Num : identifier of the goal ;*

- *Name* : name of the goal ;
- *Description* : description of the goal ;
- *Atomic* : boolean attribute that indicates if the subgoal is to be decomposed into lower subgoals.

*ACTION* : this entity describes an action appearing in the sequence of actions required to perform an atomic subgoal.

- *Num* : with the indentifier of the atomic subgoal that the action helps to perform, this attribute identifies the action ;
- *Value* : this attribute describes in a formalism chosen by the analyst the action to consider ;
- *Feedback* : the feedback triggered off by the execution of the action is described here;
- *SystemStatus* : this attribute describes the change in the system status resulted from the execution of the action.

For an example of a task description, the figure 7 presents a possible task description limited to the subgoals (without the actions) of « drawing a house » with the drawing editor of Microsoft Word97.

In this example, the main goal « draw a house » can be decomposed into four subgoals. These subgoals can further be decomposed into some subgoals. Subgoals such as C. « Select the rectangle icon » cannot be divided anymore. This subgoal contains too few actions to be decomposed again. A subgoal which contains several actions (more than 4-5) can probably be decomposed.

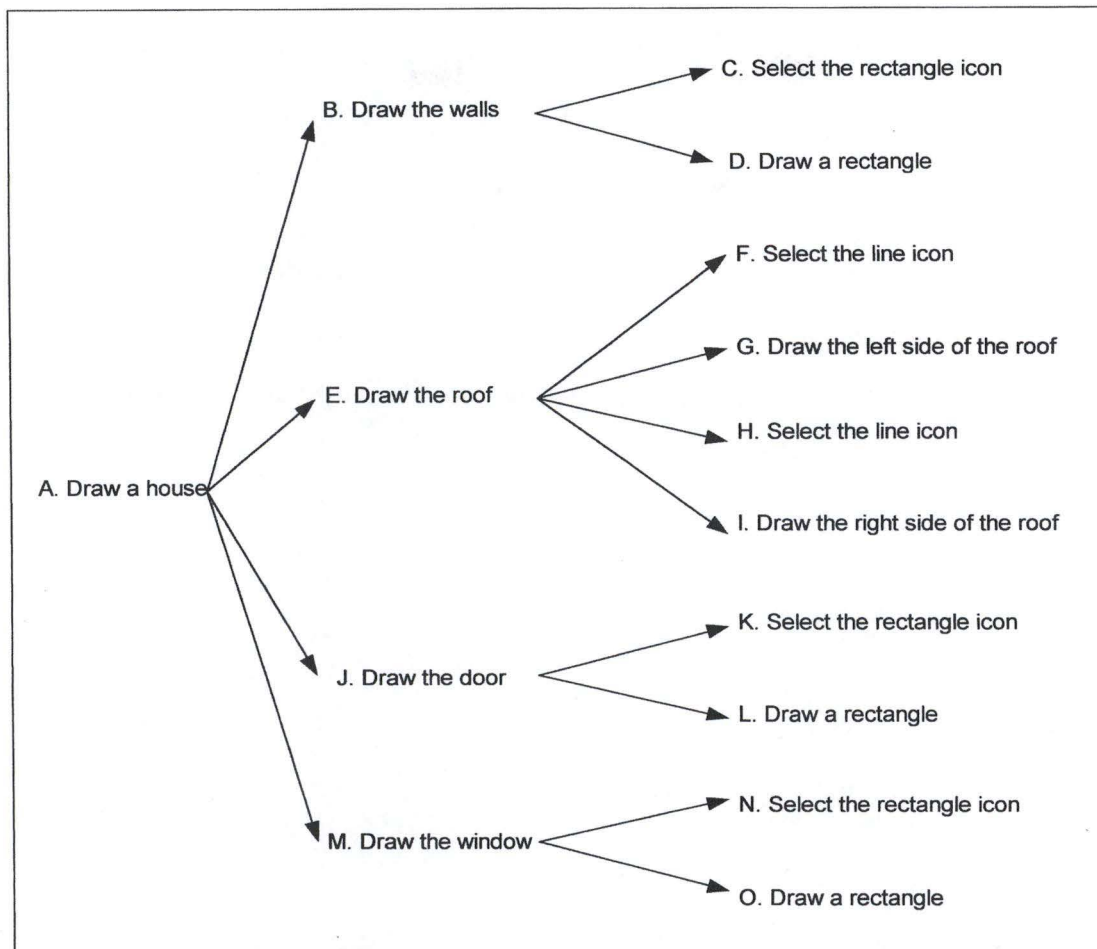


Figure 7 - Example of a decomposition of a task into subgoals

Main goal and subgoals are ordered alphabetically [LAVERY, COCKTON 97]:

- Main goal is always "A".
- First subgoal of "A" is always "B". The first subgoal of a goal is always the alphabetical next one.
- If a subgoal cannot be divided, the other same-level subgoal will be ordered alphabetically.
- When all the same-level subgoals have been ordered, the next goal to be ordered is the next unaddressed higher level goal.

This kind of graphical representation is well suited to small tasks without too many subgoals. But when considering more complex and realistic tasks, this could not be easy to perform like this.

In this case, it's perhaps more judicious to represent the task decomposition in a quicker way (but graphically less clear). Here is an example of a decomposition performed as found in [DIX et al. 93]. This is the task decomposition of drawing a matrix of rectangle by using PowerPoint97 (Fig.8).

- 0. Create slide
- 1. Create new slide
  - 1.1. Select blank presentation
  - 1.2. Select blank sheet
- 2. Insert Title
  - 2.1. Insert Title Text
    - 2.1.1. Select the Text Box Icon
    - 2.1.2. Draw the Text Box
    - 2.1.3. Type the title text
  - 2.2. Insert Title separator
    - 2.2.1. Select the line Icon
    - 2.2.2. Draw the line
- 3. Draw the demonstration stands
  - 3.1. Draw the first rectangle
    - 3.1.1. Select the rectangle icon
    - 3.1.2. Draw a rectangle
  - 3.2. Reproduce the rectangle
    - 3.2.1. Select the rectangle
    - 3.2.2. Copy the rectangle into the clipboard
    - 3.2.3. Paste the rectangle
    - 3.2.4. Place the rectangle right to make the rectangle row

Plan 3.2. : repeat [(3.2.1.)...(3.2.4.)] until the rectangles row is complete.

- 3.3. Reproduce the rectangle row
  - 3.3.1. Select the rectangles row
  - 3.3.2. Copy the rectangle row into the clipboard
  - 3.3.3. Paste the rectangle row
  - 3.3.4. Place the rectangle row above the last one

Plan 0 :

Do 1

Then do [2..3] in any order

For do other subgoals in the hierarchical order.

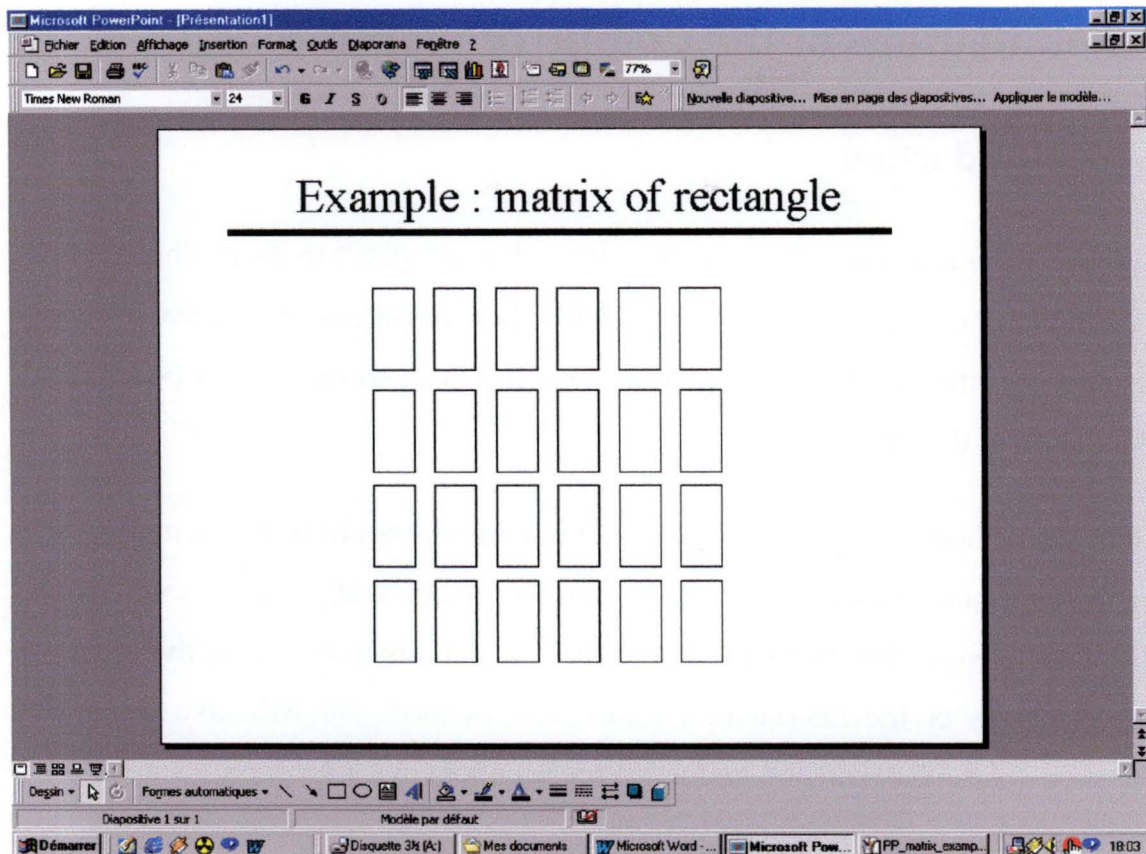


Figure 8 - The result of the task "Drawing a matrix of rectangle" in Microsoft PowerPoint97

As you have seen, we have inserted some plan indications about the order in which to perform the subgoals, to repeat them,...

We will continue to use the first one in this document, but all the examples and information given can easily be adapted to the second kind of decomposition.

Finally, we can say that the projected task is similar to the concrete task (implemented task) we have just defined by a decomposition in goal/subgoals/actions but the projected task doesn't include the actions. It's limited to the decomposition to the subgoals.

## **3. Foundations of the Cognitive Walkthrough**

### **3.1. Introduction**

It is now time to present the Cognitive Walkthrough (CW) in itself. This method includes a set of seven questions. Answering these questions may reveal usability problems in the interface. The goal of section 3 is to show how to perform an evaluation with CW.

The CW method, as an interface assessment method, should fit a task model such as the Norman model because this kind of model really makes evident the distance between the abstract task and the concrete one which is at the origin of the problems of the interfaces. To base this method, we will take the seven questions of the CW method and try to replace them in the Norman model.

The Cognitive Walkthrough method contains two different types of questions. The two sets of questions address two different fields. The first two questions are related to the goal/subgoals structure. Questions three to seven are about the sequence of actions. Before exposing all the questions and the ways to answer those, we will explain the goal of these two sets of questions.

### **3.2. Structure of the presentation**

For all the seven questions, we will present the explanation of the question, the usual answers to the question and its position in the Norman model.

#### **3.2.1. The explanation of the question**

This is simply an explanation of the question. This section will provide keys to know how answer in a good way the Cognitive Walkthrough questions.

1. Whether the analyst answers Yes, no or perhaps, he has to bring justifications to the questions. These justifications will be used as causes of the problems in the case where the answers are negative. By negative answer, we mean that the answer is either a categoric « No » or « Not always » which is equivalent to « Perhaps ». So, the negative answers include all the answers except the categorically positive ones.
2. For negative answers to questions, the analyst has to explain the cause(s) of the answer, i.e. why the analyst hasn't given an affirmative answer. This explanation constitutes a failure story, i.e. a little story which explains the assumed cause(s) of the problem. In other words,

*« the failure story explains why the analyst expects that the user will encounter a problem » [LEWIS 97].*

3. To find the cause(s) and thus to explain the failure story, the evaluator has, into others, to his disposal, the user profile and the context of work which can often explain the cause(s) of the difficultie(s). It's important that the justification to the answers related to the user profile and/or the context of work are consistent with them [LEWIS 97].
4. It's important to notice that the user profile and the context of work aren't always sufficient to justify a « negative » answer. Indeed, for example, if a user encounters a problem because he hasn't got experience of a similar application, the justification should be that « he has no experience of such an interaction technique (justification by the user profile) and that the system didn't help him providing him, for example via the interface, a clear prompt to tells him what is required to do [LEWIS 97]. So, the prompt is really used to fill in this lack of knowledge. If such a prompt is not present, the cause is not only a lack of experience but also a lack of guidance of the interface.

This is really important because when we consider the CW as part of a development of an interface, when a designer is making an interface, he has to build it in regards with, into others, the user profile and the context of work. If the user has no experience with such an interaction technique, it would be too simple to justify the first problem he would meet only by saying that he has no experience and in this situation. In this case, the only solution seems to be that the future user will have to learn it. In fact, a good walkthrough would have been to consider that he has no experience and especially that the system doesn't help him in anyway. Indeed, this is a lack of the interface and the designer has committed a mistake because he hasn't thought about the user profile, i.e. that the user hasn't experience of such an interaction means.

In the case where the user profile reveals that the future users would be at the same time experienced and no-experienced users, the interface is to be adaptable in regards with the current user, e.g. erasing the prompt when this is an experienced user.

5. Let's notice that the analyst can justify an answer « only » by a lack, a problem of interface and doesn't add the user profile and/or the context of work, e.g. where the user has experience of such an interaction mean and in spite of that, encounters a problem. Indeed, there exists causes which are inherent in the interface and which are not related to a lack of background of the user. Another example is the case where the user has no experience and the fact to have this knowledge wouldn't have hepled him and so, a prompt which would be aimed to fill in this lack of experience wouldn't probably help.

We will now present an example to clarify all that.



## *Example*

*We consider our example of the drawing editor. We consider the subgoal : « To select the rectangle icon. ». This subgoal is strange to the abstract task. Its parent subgoal is : « To draw the walls. », which is a subgoal that matches perfectly with the abstract task. We consider two different kinds of user and so two different user profiles :*

### *First user : experienced user*

*To draw a rectangle, if the user already knows at least one another drawing editor, he will probably know that he has to achieve this subgoal and how to achieve it. So, if we consider the first question in the CW method : « Will the user try to achieve the right subgoal ? », the answer will probably be : « Yes, by experience of a similar application. » because the user knows that he has to achieve it because he has already met this kind of mechanism (clicking on an icon to select a tool) in other(s) drawing editor(s). This justification constitutes here a success story, i.e.*

*« a story that explains why the analyst expects that the user will take the correct action » [LEWIS 97].*

### *Second user : unexperienced user*

*In the other side, if the user doesn't know another kind of similar application (e.g. a novice user), then we are going to consider two cases.*

*1. The system provides the user with enough help to indicate him the correct action to do. In this situation, the answer will be probably : « Yes, because the system tells him the next step to achieve his task ».*

*2. The system doesn't provide him enough help and the user can't achieve this subgoal. In this case, the answer will be probably : « No, because he has no experience of drawing editor and because the system doesn't help him ». It's important to notice that the designer has no impact on the first part of the justification (« because the user has no experience of drawing editor »). This is an assumption, an hypothesis that he*

*has to take like that. In the other side, the designer has an impact on the second part of the justification (« the system doesn't help him ») because he can improve it. This second cause is so very interesting for the designer.*

6. We could add into the justification of the negative answers (and also the positive ones) the matching between the abstract task and the concrete one.

*Example*

*The subgoal « To select the rectangle icon » is not present in the abstract task. So, if the analyst replies a negative answer to the questions related to this subgoal, he could justify by saying that this subgoal is not present in the mental model (abstract task) of the user.*

This is true but that very often, it's almost impossible to have a perfect matching between the concrete task and the abstract one (unless for the interface of direct manipulation like the virtual reality for example). So, to justify a negative answer saying that there isn't a perfect matching is only legitimate when this technical subgoal which doesn't match could have been avoided (let's remember that when we 're saying that the abstract task doesn't match with the concrete one, it's either that one or more subgoals are technical subgoals which don't exist in the abstract task or it's a subgoal which exists in the abstract task but for which the order is not the same that this existing in the abstract one).

7. There exists a dilemma. The fact of justifying an answer by the background of the user (for example, the user knows or doesn't know any similar application) is not always sufficient to justify the answer. Indeed, there exists causes which are inherent in the interface and which are also present in the similar applications. Thus, even, we can justify with his previous knowledge, this previous knowledge maybe includes some errors.

### *Example*

*In Windows, in maybe all the applications, the user can see the ItemMenu « File » in the standard toolbar. This item is used to pop-up another menu which permits the user to create a new document, open an existent one, save it ... . The name : « File » is not intuitive and is far away from the mental model of the user. The word « Document » would be more explicit and would have fit exactly with his mental model. The use of the word « File » instead of « Document » has been present in the first applications in windows and the new applications have simply followed them.*

*Here is the dilemma : If we are making a new application which requires the manipulation of files, which label for the ItemMenu concerned the designer will choose ? Either he keeps the mistake alive because the user profile reveals that the users are experienced, with a such system or application, or he decides to solve the mistake by introducing the word « Document » in his interface because the user profile reveals that the user has no experience with a such system or application.*

Finally, even the user profile reveals the experience of the user for a such kind of application, he may decide to not perpetuate a mistake even if it constitutes a habit. This last solution can be selected more if the level of standardization of the similar kind of applications is enough low.

In conclusion of this dilemma, we can say that it's important to take a critical look at similar application(s) which provides a justification to an answer (affirmative or negative).

### **3.2.2. The usual answers to this question**

For each question, this section will intend to present the most usual answers that can be given to the CW questions. Some of these usual answers come from [LAVERY, COCKTON 97], we introduce the others from our experience.

### 3.2.3. The position of the question in the Norman model

The Norman model can bring the foundations of the Cognitive Walkthrough. So, it's interesting to replace each question according to this model.

## 3.3. Subgoal-related questions

### 3.3.1. Introduction to the subgoal-related questions

The subgoal-related questions of the Cognitive Walkthrough method, the two first ones, talk about the goal structures of the considered task. It is important to recall that there are two different goal structures to take into account : the first is the abstract task and the second is the concrete one.

The reader should recall that the abstract task is the goal structure as conceived by users. It is the structure that represents the way of users that accomplish an abstract task. The concrete task is the goal structure of the task as the system requires to complete it to achieve the task. In other words, it's the goal structure of the task imposed by the interface.

#### *Example*

*When the first cash dispensers entered into action, the concrete goal structure was so : users had first to insert their credit card, then to choose the desired amount of money, to insert their secret code, to take the notes and finally their card back (Fig. 9).*

*This structure was imposed by the cash dispenser interface. But what happens in practice ? When users received the notes (that was their main goal), they often walked away and forgot to take back their credit card. The reason was that the main goal of the abstract task (which is here a real task) was completed. The solution to this problem was to force users to take their card before giving them the money. In this way the concrete and abstract task were finished together (Fig.10).*

A difference between one abstract and one concrete task may be interpreted according to a difference in the goals sequence or/and according to a different decomposition (supplementary subgoals may be imposed by the interface, ...). Another difference is the fact that the interface doesn't include all the functionalities which are present in the mental model. In this case, the degree of coverage of the interface in regards with the mental model is not totally respected. So, the interface is not totally useful.

The aim of the subgoal-related questions is to show the gap between the abstract task structure and the concrete task structure. So, for each subgoal, these two questions must be answered. But not in any order !

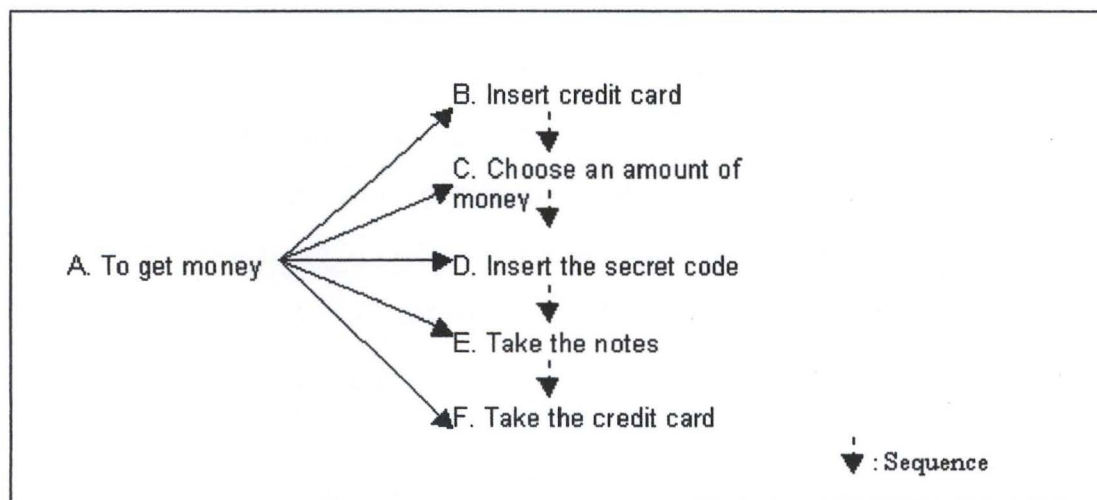


Figure 9 - Concrete goal structure of the early cash dispensers

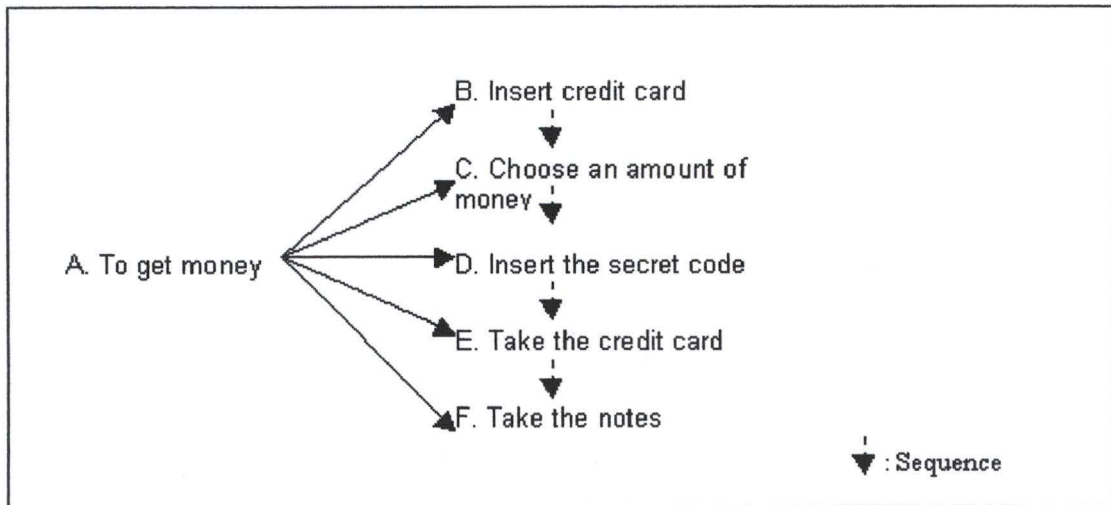


Figure 10 - Concrete goal structure of the present cash dispensers

These questions have to be asked from the main goal to the lowest subgoals following the alphabetical order. After answering these questions for an atomic subgoal, we need to answer the action-related questions according to the actions composing this subgoal. Then, we come back to the next subgoal (following the alphabetical order). The following scheme illustrates this depth-first search (Fig. 11).

The dashed arrows represent the order in which to answer questions. First, the analyst should address the subgoal-related question of subgoal B. After that, he must answer the same questions for his first successor, i.e. C. When it is finished, the analyst must focus on the action-related questions for each of this subgoal action (since it is an atomic subgoal). Then, he must go back to D which is the following subgoal. After performing the same steps for D, the following subgoal to be analysed is the next one in the alphabetical order, i.e. E.

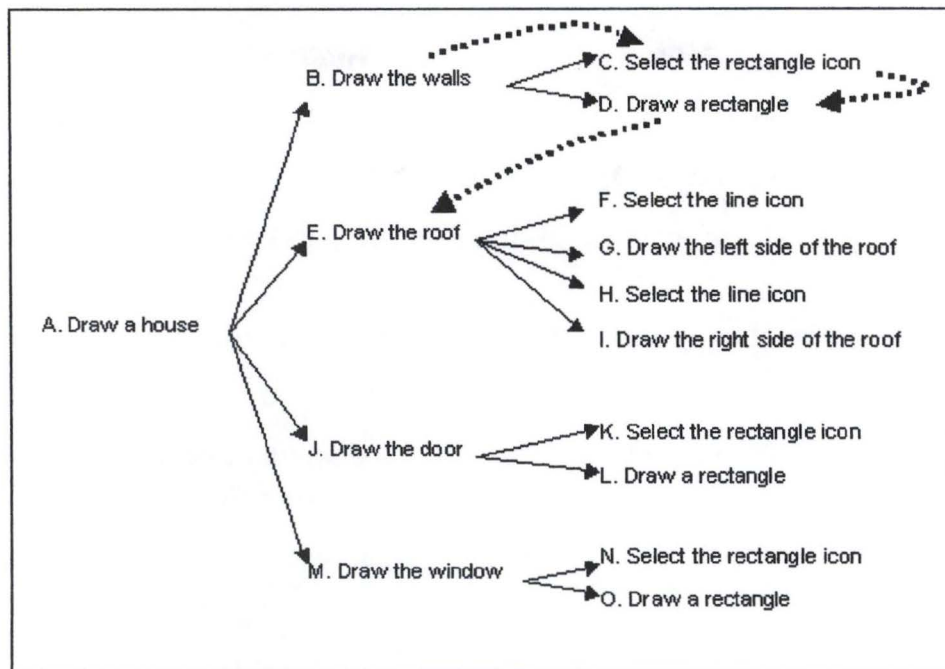


Figure 11 - Example of the answering questions sequence

The process of step-through must continue in the same way until all the subgoals and actions are analysed.

### 3.3.2. Content of the subgoal-related questions

#### 3.3.2.1. Question 1 : Will the user try to achieve the right subgoal ?

##### Explanation of the question

1. This question is related to the intention of the user to achieve the right subgoal : will the user have the appropriate subgoal in mind ?. By right subgoal, we mean the next subgoal in the concrete goal structure (concrete task). So, we want to know if users know which subgoal is the right subgoal to achieve.

This question is quite important because it permits to evaluate if the concrete goal structure (concrete task) of the task is close to the

abstract goal structure (abstract task). Note that this one is obtained from the task analysis. If the gap between the two is important, users won't probably know which is the right subgoal to achieve. But if when there is a difference between the two structures, the interface can provide information giving users help to know the next subgoal.

If the analyst gives a negative answer to this question, we can conclude that the concrete goal structure of the task is different from the abstract one and that the interface doesn't inform users about this difference.

### *Example*

*To illustrate, here follows an example. We have a drawing edition in which, to draw a shape, users have first to select the fill color before selecting the shape to draw (rectangle, diamond,...). But in real life, when people want to draw and to color a shape, they first draw the shape and then select a color and use it. So, there is an obvious difference between the abstract task and the concrete one. In this case, it is likely that users will have a problem because they will try to select a shape tool first. The analyst will give a negative answer to this first question.*

2. But supposing that when users try to select a shape tool, the interface provides a feedback (i.e. a hint text) saying to users that they have to first select a fill color. Now, the interface provides information about the concrete goal structure and the next subgoal to achieve. The analyst will give thus, in this case, an affirmative answer.

Furthermore, users can know about the right subgoal to achieve, even if no information is provided and the concrete structure differs from the abstract one, for instance because he is used to the task because he is a former user of a similar system.



3. Supposing that the analyst answers « No » to this question. This implies that he thinks that users won't try to achieve the right subgoal. Thus, it is likely that that users won't succeed in performing the task. The analyst may then stop the walkthrough as he found a problem that make the realization of the task impossible. This is a wrong choice.

Whatever the answer (affirmative or negative), the analyst must go on with the hierachically lower subgoals because there are perhaps problems in those, not only related to the goal structure but also related to the actions. As it is important to find out the possible other problems in the lower subgoals, the walkthrough must be continued.

However, to answer the Cognitive Walkthrough questions related to the subgoal-children of the one considered, the analyst will have to assume that he didn't give a negative answer but an affirmative one. Because the fact that there is a problem at some point in the goal structure doesn't have to influence the rest of the analysis.

#### **Usual answers to this question**

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar system.
- Because the system tells the user, for example through a modal dialogue.
- By experience of task on a previous system or another system.

- Because users have the knowledge to achieve a higher subgoal (inheritance from subgoal X question 2).

Inheritance (See 3.3.2.2. for a complete explanation) is a way to answer this question that we introduced to shorten the time needed to perform the method, when it is possible. In the beginning, we didn't use the inheritance and we answered at this question using a bottom-up approach. We have always delayed the answer to the lower level of actions, i.e. we only gave an affirmative answer to question 2 for a certain subgoal if we could give an affirmative answer to the question 2 of all the subgoal lower subgoals.

- By experience of the abstract task.

Assuming that users know the abstract task subgoal hierarchy, if the concrete subgoals hierarchy implemented in the interface is the same at the subgoal considered, it can be said that users will try to achieve the right subgoal of the concrete task because it is the same that he would try to achieve while performing the abstract task.

It is important to always justify an affirmative or a negative answer, especially when answering one of the above standard answers. The analyst must say in what the experience is similar or how the system tells users about the next subgoal, and so on. The note applies to the answer of all the other questions of the Cognitive Walkthrough method.

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Because there is a too important gap between the concrete goal structure and the abstract one (and because the system doesn't provide information to fulfill this gap).

- Because the user has no knowledge of the task on the system and the system doesn't provide the information which would indicate which subgoal to try and achieve.

### **Position of the question in the Norman model**

The Norman model puts the distance between the abstract goal structure and the concrete one in evidence. The first question in the CW is goal-related. We have two possibilities: either the subgoal matches perfectly with the abstract goal structure, then there is no problem, or the subgoal doesn't match perfectly with the abstract goal structure and then this question will evaluate if the distance between this subgoal and the abstract model of the user is important or not.

After executing a sequence of actions, perceiving and interpreting its results, users have made an iteration through the Norman model. Indeed, this model lets us express the task in term of a hierarchy of goals/subgoals. So, when conscious of the now current state of the system, users will select the next subgoal to achieve to perform the next iteration in the model.

All the first three points of the Norman model could be covered, including goal setting, intention formation and the specification of the actions. Only the first step of the specification of actions (the translation into desired states of the system) is relevant for non atomic subgoal. For atomic subgoals, the three points of the specification of actions intervene.

The matching between the abstract and concrete task is evaluated here. Let's notice that if the concrete subgoal perfectly matches with the abstract one, this crossing is very little and so very easy.

3.3.2.2. *Question 2 : What knowledge is needed to achieve the right subgoal ?  
Will the user have this knowledge ?*

**Explanation of the question**

Whereas the first question addresses the problem of knowing if users will try to perform the subgoal they have to, this question is about the knowledge to achieve the subgoal. There is always some knowledge on how to realize a task or a part of it. This question forces the analyst to determine which knowledge is necessary, because users may know which subgoal to achieve but not how.

Secondly, if the user has to achieve a subgoal, we have to know whether he has now the knowledge or whether he will have the required knowledge in the future, for example because the system gives him the information needed to know how to do.

If an affirmative answer is given, the cognitive walkthrough can go on. If a negative answer is given, the analyst has to know exactly which part of the knowledge users won't get and why. In other words, we want thus to know at which lower subgoals or actions, the system has a lack of information to complete the subgoal.

**Usual answers to this question**

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar systems.
- Because the system tells the user, for example through a modal dialogue.

- By experience of the task on a previous system or another system.
- Because users have the knowledge to achieve a higher subgoal (inheritance from subgoal X question 2).

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Because users have no experience of either the system or the task and because the system doesn't provide complete information.
- Because users haven't the knowledge required to perform one of the lower subgoals.

In this case, the answer of this question is deferred until the analyst gives a negative answer to a question 2 of a subgoal-children (bottom-up approach).

For atomic subgoals (i.e., pre-terminals in the goal tree), this answer may be deferred to the action-related questions. Users don't have the knowledge but perhaps the system gives him the necessary information. That's why a closer look in the sequence of actions of this subgoal can help to answer its question 2.

### **Inheritance**

The last answer may be often used to answer this question. What's why we will explain more about this.

If we answer at this question that the user has the knowledge, all the sub-subgoals of the previous subgoal must be answered in the same way at this same question and also at question 1. We call that "inheritance" (top-down approach). Let's notice that the answer at action-related questions of all the actions of all sub-subgoals of the previous subgoal seem to be "yes" (by inheritance). It's a solution to

reduce the burden of the Cognitive Walkthrough because it permits reduction of the number of questions to answer and justify. Thus, we perform a top-down approach here in contrast of the bottom-up approach. The bottom-up approach delays the answer to question 2 to the analysis of lowest subgoal of the goal structure. In that approach, if users have all the knowledge of the lowest subgoals, we can say that they have knowledge for the higher ones.

In the top-down approach, when we can give affirmative answer to a question two of a subgoal, we can assume that the rest of the goal tree that is dependant on this subgoal is satisfied. It is less taxing.

#### *Example*

*The example (Fig.12) is taken from a diagram modification task. We will only consider the subtask of selecting the lower part of the diagram as shown in the following figure.*

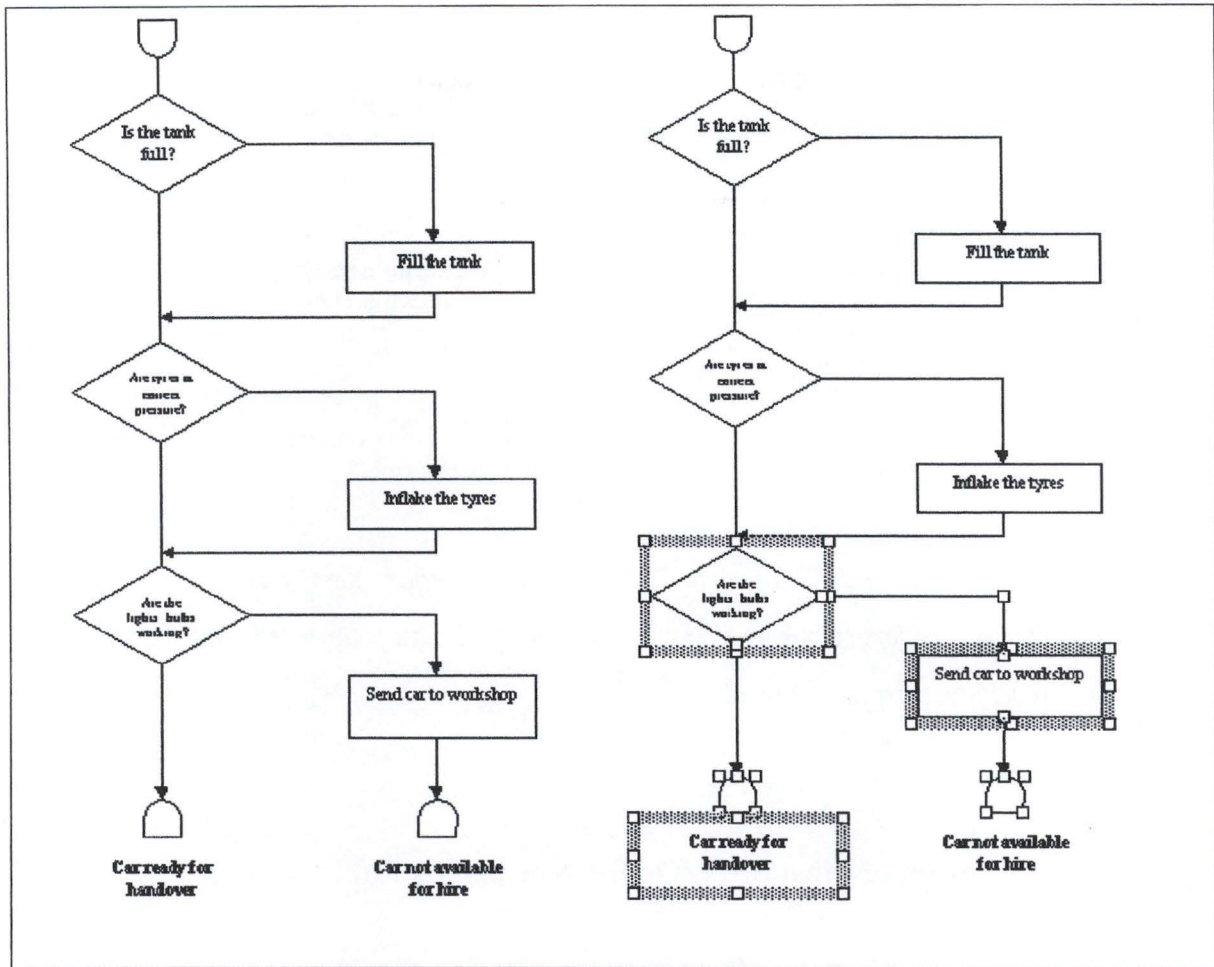


Figure 12 - Example of a task selection with Microsoft Word97

The goals/subgoals decomposition of this subtask is illustrated by Figure 13.

For the considered subgoal C, if users know how to perform multiple selection in the drawing editor of Microsoft Word, the analyst will give an affirmative answer to the question 2. Then, it may be viewed as a waste of time to perform the method for the lower subgoals D and E because the analyst knows that users have the knowledge to achieve the subtask of selecting the part the diagram, and from this, they also have the knowledge of selecting the « Select Objects » (permits to switch from the text edition mode to the objects selection mode) icon and of dragging a rectangle.

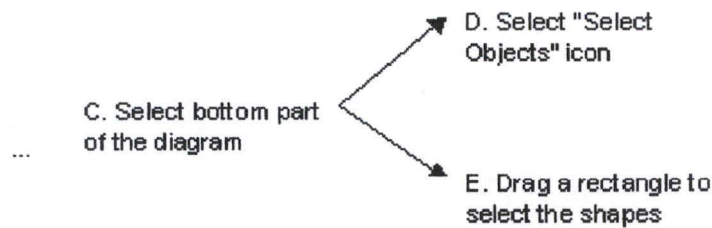


Figure 13 - Goals/subgoals decomposition of the subtask :  
selecting the lower part of the diagram

*In conclusion, the inheritance can reduce time and effort for cognitive walkthrough analysis, as answering questions for certain subgoals can be avoided since users know how to achieve a upper subgoal.*

### **Position of the question in the Norman model**

This question hasn't an explicit place in the Norman model for the non-atomic subgoals. For those, this question is only a way to reduce the cognitive walkthrough analysis.

For the atomic subgoals, this question addresses the knowledge of users according to the sequence of actions to realize and perform the subgoal. So, the question is clearly located in the third step of Norman model, i.e. the « specification of actions sequence » where in relation with his intention, the user has to translate it into a sequence of actions that would permit to complete the current subgoal. The second and third step of the specification of actions sequence, « to find out the devices of the mechanics of the interface which will produce this state » and « To find out the required manipulations of these mechanics », are concerned by this question.



## 3.4. Action-related questions

### 3.4.1. Introduction to the action-related questions

After answering subgoal-related questions for an atomic subgoal, the analyst may step through the sequence of actions that compose this subgoal. Indeed, at this level, a subgoal isn't composed of a set of lower other subgoals, but is composed of a sequence of actions that must be completed in a specified order to achieve the subgoal.

### 3.4.2. The action-related questions

#### 3.4.2.1. *Question 3 : Can the user perceive that the correct action is available ?*

##### **Explanation of the question**

This question is perception-related : can the user know that a widget exist that will allow them to perform what they need to ? We want to know if the interface is made in a way that permits users to perceive the correct action to select. The correct action to realize an atomic subgoal to realize is the next action of the task sequence.

When answering this question, the analyst has to specify the interface mechanisms that would be used to, or that would be perceived as a way to, achieve the action.

Finally, let's remark that the question was initially : « Will the user perceive that the correct action is available ? » but we think that the word « Can » is more appropriate to indicate a perception. Indeed, this question is perception-related.

### Usual answers to this question

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar systems.
- By experience of the task on a previous system or another system.
- Because users have the knowledge to achieve a higher subgoal (inheritance from subgoal X question 2).

As explained in section 3.3.2.2., an affirmative answer can be given at any question related to a subgoal or an action if the analyst has given an affirmative answer to the question 2 of an upper subgoal.

- Because the system provides the information required (e.g. : an icon representing the action)

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Because the information provided is misleading (e.g. a wrong-labelled button may prevent users from understanding the action it triggers off).
- Because the system provides too much information about many different actions (e.g. an interface where lots of actions are available through the use of icons ; in this case, there are perhaps too many icons, so users won't pick up the one they need).
- Because the action is not easily accessible ; it requires too many interface manipulations to trigger off the action.

This may be the case, if users have to go through two or three levels of a pop-up menu, and/or the information (e.g. the menu items labels) is no directly related to the actions.

### **Position of the question in the Norman model**

This question is only about perception of the correct action in the sequence of actions to achieve an atomic subgoal. In the Norman model, we can replace this question in the action specification part.

Since this question is about the perception of the correct action to complete to progress in the realization of the current subgoal, it allows examination of whether users can determine the mechanisms of the interface that would product the desired states of the system and to determine the manipulations to realize on these mechanisms. So, the second and third step of the specification of actions are related to this question.

#### **3.4.2.2. *Question 4 : Will the user associate the correct action with the subgoal they are trying to achieve ?***

##### **Explanation of the question**

When an action is perceived, users will only make it if they see that there is some connection between this action and the subgoal they want to achieve. An user will associate an action to a subgoal if he thinks that this action will change the present state of the system into his desired state of the system.

For example, in a drawing editor, if a user wants to draw a rectangle and if he notices the rectangle icon in the drawing toolbar, he will probably associate the action « Click on the rectangle icon » with the subgoal « Draw a rectangle » because there is an obvious link between the action and the objective to achieve.

### Usual answer to the question

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- Because the system provides the information required (e.g. : a text in the status bar).
- By experience of the task on a previous system or another system.
- Because users have the knowledge to achieve a higher subgoal (inheritance from subgoal X question 2).
- All other actions or choices look wrong.
- If there is some connection between the action and the subgoal they are achieving.
- Answer deferred to question 6.

This last answer will be explained further. The analyst can sometimes defer the answer to this question until question 6. Sometimes the user is not sure about the action to perform and will make a few attempts to see the system response. According to this response, he will go on performing the action. For instance, the user notices an icon but he is not sure that this icon will provide him what he needs. He probably will move the mouse pointer to this icon and, seeing the feedback (the hint text), he will know this is what he was looking for (or not). This is an information feedback. Here follows an illustration (Fig.14).



Figure 14 - The hint text of an icon

So, sometimes the representation of the action or the information, provided by the system, doesn't permit users to be sure about the association of the action with the current subgoal. But the feedback can provide more information that confirm the connection.

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Answer deferred to question 6.
- Because the system doesn't provide information or this information is not unambiguous.

The system can provide information but this may be not useful if users don't understand those. For instance, if a button is wrong-labelled, users may not associate this label with the action they want to accomplish.

#### **Position of the question in the Norman model**

Question 4 is related to the action sequence specification part of the Norman model, step one, i.e. «To translate the objectives and intentions to desired states of the system ». The specification of the action sequence involves the translation by users of their objectives into desired states of the system. Then, they have to determine the mechanisms of the interface which would product these states.

It seems that these two first steps are treated by question 4 : « Will the user associate the correct action with the subgoal they are trying to achieve ? ». With a given action, we want to know if users are able to determine that this action is related to the objectives they want to achieve.

Answering this question, according to the Norman model, permits to find out if the interface offers ways for users to translate their mental goals into interface mechanisms.

#### 3.4.2.3. *Question 5 : Will the user perceive the feedback ?*

##### **Explanation of the question**

This question is perception-related. After the execution of the action, the interface often provides feedback that informs users about the effect of the action on the system.

Thanks to the feedback, users are aware of the success or the failure of the sequence of actions to reach the desired state of the system. The feedback is shown by a set of physical variables of the system. So, this question forces the analyst to look after any interface means and to ask himself if users will notice any feedback.

##### **Usual answers to this question**

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar systems (by conscious attention : users know that a feedback will be provided and they look at it).
- Users know that the interface will provide a feedback because they have already used this system or a similar one and they consciously look after it because it will inform them about the result of the actions on the state of the system.
- If they are focusing on the area of the screen which provides the feedback.

- If the feedback can not be ignored, for example audio or a message box where users have to click on the "OK" button before doing anything else.

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Because there is no feedback.
- Because the feedback appears in a part of the screen where users' attention is not focused, or the appearance of the feedback brings little difference with the previous state of the interface (without feedback).

#### *Example*

*The following figures illustrates this last point. The task considered concerns the filling of a form in a Web page. Users fill in the different text field (see fig. 15). When users select the e-mail text field, the message shown in the status bar changes to indicate the syntax of the e-mail address to enter (fig. 16).*

*But as this feedback appears in a part of the screen, the status bar, where the attention is not only rarely focused but also is not focused in this case because users are likely to look at the form. Because the human attention is selective such a feedback will be almost never noticed (fig. 16).*

Name  First Name   
 Club  Licence Nr   
 Ranking  Sex   
 Email  Phone

**Categories**

	A	B1	B2	C1
Men Single	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ladies Single	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	A	B	C	
Men Double	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

Terminé

Figure 15 - The state of the web browser before clicking in the Email text field

### Position of the question in the Norman model

This question is obviously related to the step 5 of the Norman model, i.e. the perception of the state of the system. Following the manipulation of the system mechanisms to execute the sequence of actions, the state of system has changed. This change is visible thanks to the physical variables of the system and if it is perceived by users.



Name  First Name

Club  Licence Nr

Ranking  Sex

Email  Phone

Categories

	A	B1	B2	C1
Men Single	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ladies Single	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	A	B	C	
Men Double	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

Please enter a valid email address(like verriers@badminton.be)

Feedback area

Users' attention area

Figure 16 - The state of the system after clicking in the Email text field

#### 3.4.2.4. Question 6 : Will the user understand the feedback ?

##### Explanation of the question

When users have performed a set of actions on the interface and since a feedback was provided to them by the system, the question is to know if users will understand this feedback.

It is important for the analyst to ask this question, because a misunderstanding of the feedback can lead users not to understand the new state of the system, to believe that the system has not performed what they were waiting for after the execution of the sequence of actions, and so to force users to make some other

sequences of actions whereas the current state of the system is the one required by users.

### **Usual answers to this question**

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar systems.
- If users are familiar with this system or similar ones, he knows about the feedback provided for the current action and so, he will recognize it.
- If the feedback is unambiguous.

An unambiguous feedback is a feedback that will be clearly understood by users. For example, if users want to perform a forbidden action, the system may react, prevent users from doing this action and as feedback, give an alert dialog box where information about the forbidding of the action and its reason is given.

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- If the feedback is ambiguous.
- Because there is no feedback or users don't perceive it (from question 6).

### **Position of the question in the Norman model**

The step 6 of the Norman model is about the interpretation of the new state of the system (got by the execution of the sequence of actions). Users have to translate the physical state into mental state and to interpret the perceived state in relation with their intention. For

instance, in a word processor, he has to translate the physical state « darkened text » into the mental state « selected text ».

Indeed, the question 6 is about this step of the Norman model as we want to know if users understand the feedback perceived, i.e. if the interface language is compatible.

**3.4.2.5. Question 7 : Will the user see that progress is being made towards solution in relation to their main goal and the current subgoals ?**

**Explanation of the question**

The analyst has to think instead of the user to find out when an action has been made, if there has been progress or not (from users' point of view). We could say that when an action has been made, and when the user knows how to perform the task (i.e. he has the knowledge to do this), there is progress in the subgoal which includes this action and progress in all super-subgoal which includes this subgoal to the main goal.

So, to answer this question, the analyst has to decide that there is a progress when this is a significant progress seen by users.

**Usual answer to this question**

When the analyst gives an **affirmative answer** to this question, the most common answers are usually the following ones :

- By experience of the system or similar systems.
- Because the system provides him with the information.

When the analyst gives a **negative answer** to this question, the most common answers are usually the following ones :

- Because there is no feedback and users have no experience of the system.
- Because users have not understood the feedback (from question 6).

### **Position of the question in the Norman model**

The base of this question is step 7 of the Norman model, the « evaluation of the state of the system ». At this point, users have to compare the perceived state of the system in relation with the desired objectives. Users want to know if, after executing a sequence of actions, they have achieved their objectives or are closer to these.

Question 7 is clearly about this problem because it forces the analyst to ask himself if users see that there is progress in completing the objectives.

## **3.5. Conclusion on the foundations of the Cognitive Walkthrough**

Given the comprehensive matching with the Norman model, the CW permits to evaluate the importance of the distance between the abstract task (his mental representation of the task he has to achieve) and the concrete task imposed by the interface. This evaluation will be able to be performed at the execution and evaluation levels.

## **4. Conclusion**

This chapter has brought foundations of the Cognitive Walkthrough method and has explained how to perform such a method. The performing is the answering to the questions of the Cognitive Walkthrough without forgetting to justify their in the aim to find causes of the problems. Going through the questions only reveals possible usability problems but says nothing about the seriousness of the problems and possible failure in the task realization. This will be assessed in the fifth chapter.

LOCATION OF THE COGNITIVE WALKTHROUGH METHOD IN THE  
TASK ANALYSIS

## **1. Introduction**

This chapter is aimed to present you the location of the Cognitive Walkthrough in a design development process. As design development process, we have selected the task analysis approach developed by Professor Bodart [BODART 98].

First, we will present an overview of this task analysis method and after that, the two possible locations of Cognitive Walkthrough within this approach.

Secondly, we will present the design criteria approach to user interface design. This also influences the way designing an interface and we will compare a technique using a kind of criteria (Heuristic evaluation) with the Cognitive Walkthrough. Finally, we'll briefly present Heuristic Walkthrough, a method that combines both techniques.

## **2. Review of the task analysis**

We consider here task analysis as part of a general lifecycle for developing an interface. of the approach is as taught by F. Bodart for the second master's degree ([BODART 98]).

### Example

To give examples of our future explanations, we will introduce you a case. It is about the drawing of a house in a drawing editor that we have to create. Let's notice that, in this perspective, this drawing editor will include the possibility to make a house and other objects which is easily assimilated to it but won't include all the functionalities of a complete drawing editor. The aim is here to show how to develop an interface and so we assume that the interface to build is a reduced drawing editor which only includes the mechanisms to draw objects like house.

## 2.2. The inputs of the task analysis

The first step in the task analysis is to analyse the abstract task (which is here a real task) watching the users, the workstation and the knowledge of the domain (Fig.17).

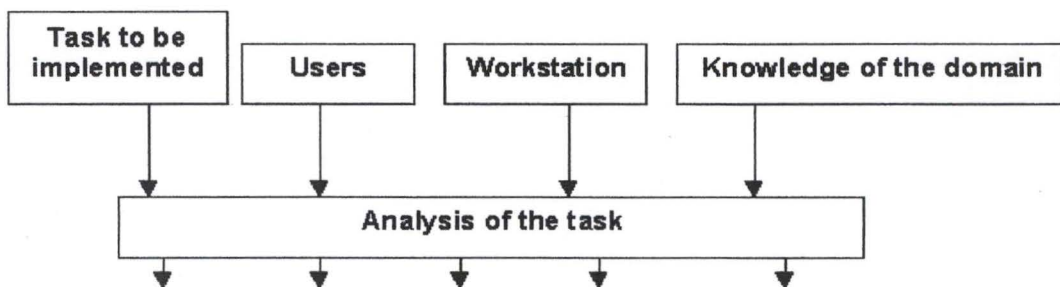


Figure 17 - The inputs of the task analysis

### The users

It's important to know what kind of users will perform the implemented task because they represent the potential future ones who will use the interface and they will have an influence in the design of the future one.

*Example*

*In our example, the user may be a secretary, an office employee, ... who makes a drawing of a house on a sheet.*

**The workstation**

The workstation represents the environment of work where the task is achieved.

**The knowledge of the domain**

It's the knowledge of the abstract task.

*Example*

*The knowledge of drawing a house with a pen on a sheet of paper.*

**2.3. The results of the task analysis**

This analysis of the task will produce several results (Fig.18).

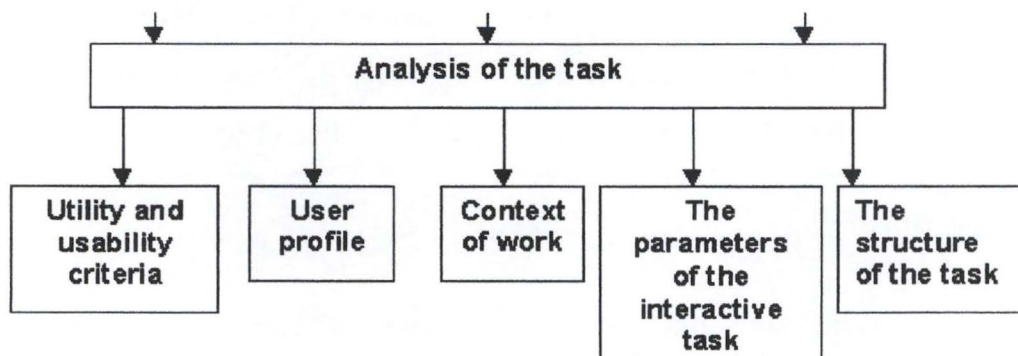


Figure 18 - The results of a task analysis, following Pr.Bodart's approach



## **The balance of each of the utility and usability criteria**

According to the analysis of the task, we'll give more weight to some criteria, in regards with their importance and the context of use.

### *Example*

*In our example, the subjective satisfaction of the user must be high to permit user-friendliness for the user and the degree of coverage of the mechanisms of the interface has also to be high because it would be good if the functions of the interface can cover the most important functions desired by the user. This would permit a drawing closer by the mental representation of the drawing the user has, even if we 're building a « reduced » drawing editor.*

*Then, the learning time of the task can be long because we can assume the user will often have to draw diverse drawings. If the user only performed the task rarely, the learning time should be very short because it would be inadmissible to constrain the user to pass too much time all learning again every time he will perform the task. Let's notice that the balance of each of these criteria are dependent on the importance and context of the task revealed by the task analysis.*

Thus, the different weights on each of the criteria that we have assigned here are one possible balance of these criteria assuming a specific task but according to another context of the task, another balance could have been chosen.

## **The user profile**

The user profile describes the future stereotypes of users of the interface according to the potential users we have considered. In fact, we're trying to infer some characteristics of the users of the future interface. The user profile will reveal the experience of the abstract task, the motivation of the user and moreover, the experience of the computer system, similar or not with the future

interface and the experience of a complex interaction mean. Moreover the knowledge of the same kind of concrete task on a similar application must be specified. Let's notice that there exists different ways of analysis such as the interview or the direct observation to obtain this information.

### **The context of work**

We have to include the context of work in the design of an interface. For example, the screen of an on-board computer for a GPS system will have to be little according to the place available in a car. In this situation, the mechanisms provided by the interface must be appropriate in regards with this specific work environment.

However, for generic software like a word processor and a drawing editor for example, it is important to take in consideration the different possible environments of work in the design of the interface. Some are maybe noisy and users may have big screens, others may be quiet and users may have little screens and perhaps perform other manipulations in the same time.

The analyse of the workstation will reveal :

- The physical environment where the task will be performed, i.e. :
  1. The conditions of work : visibility, noise, stress, ...
  2. The environment (surroundings)

#### *Example*

*The environment is noisy and little luminous. The conditions of work are good.*

- The task allocation (person, function, role)

- The single or multi-task allocation, which will reveal if during the task execution with the interface, another activities are performed or not.

### *Example*

*The secretary maybe has to answer phone and make phone call performing the task of drawing.*

### **The parameters of the interactive task**

The four first products will permit the analysis of the ergonomic specifications which will reveal the interaction style and the dialogue attributes<sup>3</sup>.

### **Task Structure**

This is the decomposition of the task in a structure of goal/subgoals. The first notice we can put in evidence here is to know if it is the decomposition in goal/subgoals of the abstract task or the projected task. The answer is that we need the both in a purpose to compare their.

Performing the decomposition in goal/subgoals for the abstract task is rather easy. Here is the decomposition in goal/subgoals of the abstract task : « Drawing a house » (Fig. 19).

---

<sup>3</sup> See [Bodart 98] for more explanations.

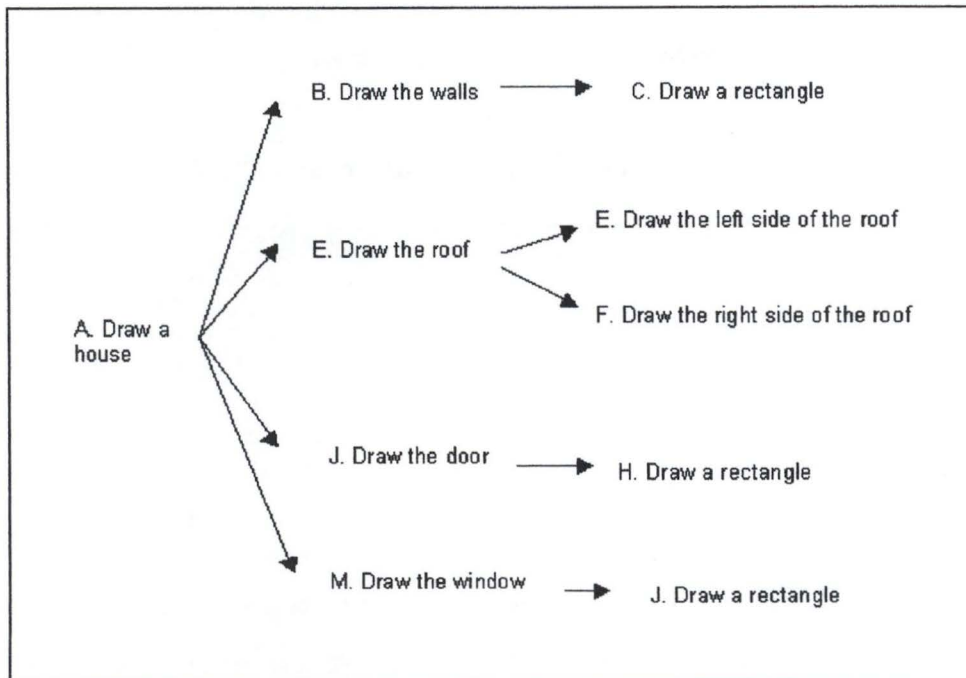


Figure 19 - Task description of the abstract task « Drawing a house » on a sheet of paper (abstract model of the task perceived by the user)

Performing the decomposition in goal/subgoals for the implemented task (i.e. in the future interface) is not yet possible here. Indeed, at this point, we can only describe the decomposition in goal/subgoals for the projected task. To this aim, we'll use a decomposition which resembles the structuring model of the operational knowledge (TKS model : [JONHSON 92]). This decomposition will permit a better integration of the CW in the task analysis as part of a global walk of creation of an interface. The TKS model of the task can be useful to know how to decompose the main goal into lower subgoals. But our decomposition is a bit different in the sense that, in the decomposition we propose here, only the atomic subgoals are composed of actions.

Moreover, we give to « action » a different meaning than in the TKS model. We have defined what we mean by « action » in Chapter 3, section 2.6.

In our example, here is the decomposition of the task in goal-subgoals as result of the analysis of the task.

We have a goal, which is an objective for the user, in our example the goal is : « To draw a house ». The subgoals related to this goal are :

1. To draw the walls.
2. To draw the roof.
3. To draw one windows and one door.

These subgoals match with the abstract task. The user knows that to draw a house, he has to draw the walls. This subgoal is present in the abstract task (a real task here) of drawing and also in the projected task. Let's notice that these three subgoals can be performed in any order. If the interface imposes an order different from the abstract structure of the user, there won't be a matching because, when drawing a house on a sheet of paper, users can first start the drawing with any part of the house.

Now, we will only focus on the first subgoal : « To draw the walls. », we assume it can be divided in two subgoals :

1. To select the rectangle icon.
2. To draw the rectangle using the rubber-banding method.

### **First Remark**

The reader could wonder if, for example, « To draw the rectangle using the rubber-banding method » is a subgoal or an action. In fact, it's a subgoal because this one can be decomposed into more elementary elements that we called actions in regards with the definition we gave of this term in chapter 3.

What we mean by action is : an action consists in an operation executed by users via an interaction means which have an impact on the interface and/or on the state of the system.

### **Second Remark**

The rubber-banding method is a technique which supports easy drawing of rectangles. We 'll give you an example of rubber-banding in the drawing editor of MS Word 97.

#### *Example*

- 1. After having selected the rectangle icon, select the corner of the future rectangle from which you want to start and press the mouse button. We assume it's the upper left corner of the rectangle.*
- 2. Keeping pressed the mouse button, slide the mouse pointer towards the lower-right corner of the rectangle you want to draw.*
- 3. When you have the desired size of the rectangle, you have to release the mouse button.*

This technique is also used to select a set of files or icons for example.

### **Third Remark**

The subgoals we have chosen already imposed mechanisms of the interface even we aren't yet at the implemented task level. Indeed, the projected task level is similar to the implemented task level as far as subgoals are concerned. But the actions aren't yet present at the projected task level.

The subgoal 1 « To select the rectangle icon » is imposed by the interface. It doesn't match with the abstract model of users. In fact, this subgoal doesn't exist in the abstract model of the user. Indeed, the user knows that he has to draw the

walls but to perform it, he has to translate his mental state into desired states of the interface. The same remark can be made for the second subgoal.

The second subgoal « To draw the rectangle using the rubber-banding method » is closer to the abstract task where the user also has to draw the rectangle. However, the way to draw the rectangle is a bit different in the projected task than in the abstract one.

What is most interesting here is that we really see the move from the abstract task to the projected one. As much of the interface introduces subgoals that don't match with the abstract task (excepted for the interface like the virtual reality, i.e. interface where the direct implication is very developed), the challenge of every interface is really to allow the user to perform this crossing easily, i.e. requiring a few effort by the user to reduce this distance between these two worlds.

So, when we are speaking about projected subgoals not matching the abstract ones, at this moment, the user, to perform his subgoal, for example : « To draw the walls. » , will have to translate this last one into desired states of the system (the two subgoals seen above).

Thus, he has to pass from a mental world to a physical world. This gap is specified in the Norman model. The user expresses his subgoal : « To draw the walls. » in significant terms for him, in regard to the mental representation he has about the problem (psychological terms). His aim is to translate this subgoal with the help of the mechanisms of the interface expressed in physical terms. Similarly, throughout interaction, the user will have to evaluate the achievement of his subgoal.

We can notice that the model of the operational knowledge (that we used to perform the decomposition goal-subgoal in a little modified way) has already imposed features of the future interface. Indeed, we have defined certain

subgoals which already impose some mechanisms on the future interface. In fact, we remind you that we're building an interface and for example, when you 're speaking about the subgoal : « Select the rectangle icon », we are already speaking about physical mechanisms of the future interface even though we don't speak yet about actions and interactive concrete objects.

The reasoning is : « The subgoal of the user is ... ». If the subgoal matches with the abstract goal structure of users, there is no problem, the good sense and his knowledge of the domain (his knowledge about the abstract task) will permit to the user to think about it. If the subgoal doesn't match with the abstract goal structure, the user will have to pass with more difficulty from mental terms to physical terms imposed by the interface. The model of the knowledge of the control, the Norman model, can help us to develop an useful and usable interface in a way we will try to build the interface reducing the distance between the abstract and concrete (or projected) task. This reduction is performed to satisfy utility and usability criteria. In this way, this model of Norman can validate (or refute) the model of the operational knowledge, the TKS model. Here is the entire decomposition in goal/subgoals of the projected task : « Drawing a house » (Fig. 20).

The subgoals in red are the subgoals which don't exist in the abstract task. The red-boxed subgoals are the subgoals which don't exactly match with the abstract task. We have specified that the order of subgoals B, E, J and M was able to be performed in any order to match better with the abstract task. If we suppose the interface had imposed this order, this way imposed by the interface to draw a house doesn't fit necessarily with the abstract task.



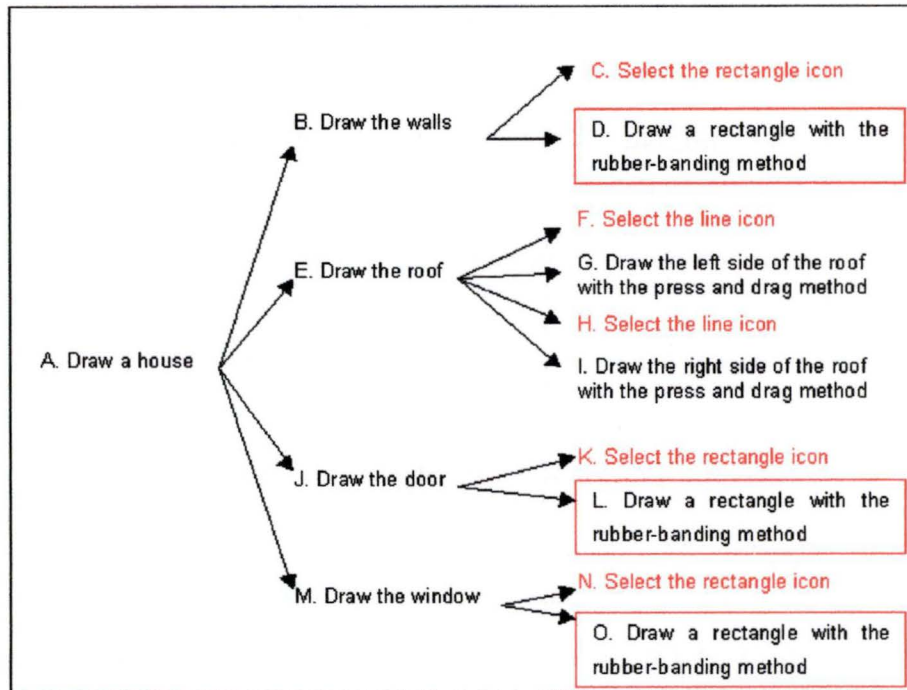


Figure 20 - Task description of the projected task « Drawing a house » in a drawing editor (concrete model of the interface)

The subgoals G and I use a method, a technique which is very close to the abstract subgoal. Indeed, when you want to draw a line with a pen on a sheet of paper, you have to put your pen on the sheet (this action is represented in the concrete model in a press of a mouse button) and after that, you draw a line keeping your pen on the sheet (represented by moving the mouse, keeping the mouse button pressed). These two subgoals are very close to the abstract task.

Be careful, let's notice however that it's not bad to have concrete subgoals that do not match exactly with the abstract ones or even do not exist in the abstract goal structure. It's almost inevitable (excepted for the interface like the virtual reality, i.e. interface where the direct implication is very developed as said earlier). For example, the rubber-banding method doesn't fit exactly with the abstract goal structure of the user where to draw a rectangle, the user takes a pen and a rule, puts their on the sheet and begin to draw the first side of the rectangle, then the

second side and so on. Unfortunately, this way seems to be complicated to achieve on an interface with a mouse or even a light pen. The rubber-banding helps the user to draw easily a rectangle and after clicking on the rectangle icon, information appears in the status bar to explain how to use this way of drawing rectangle (even if we can discuss this utility of this status bar because it is not perceived easily by the user, as it is not in the users' attention area and it provides sometimes poor and bad information).

The three products of the analysis of the functional specifications are :

1. Specification of the information.
2. Specification of the functions.
3. Chaining graph.

These three results lead to a use case. We won't develop these points here because these are not relevant for our discussion (see [Bodart 98] for more explanations).

Finally, the result of the global walkthrough of the task analysis permits to design the interface.

## **2. Location of the Cognitive walkthrough in the task analysis**

Given the comprehensive match between CW and both the Norman model and as the above approach to task analysis, creating similar structures for operational knowledge, the CW method could be used in the structuring of the task, i.e. in the decomposition in goal-subgoals of the projected task in regards with the

abstract task. Indeed, we have seen that the structure in goal-subgoals already induced features (physical mechanisms) of the interface and that we could use the Norman model to validate or refute this structure. In other words, we can already think about to reduce the distance when we make the structure in goal-subgoals.

We could already use the CW method here. And this, to wonder if the projected goal structure is a good matching of the abstract goal structure of the user. The question 1 : « Will the user try to achieve the right subgoal ? » will let us evaluate this matching. Unfortunately, this use of the Cognitive Walkthrough will stop at questions related to the subgoals. Indeed, we are at the projected task level and we have not yet the actions.

Let's notice that we have to modify a little the inputs required to perform this « reduced » CW.

First, we need not any more necessary an interface or even a prototype. That's all to the good because at this moment during the design development process, we haven't yet a such interface or prototype.

Secondly, we need now the projected task instead of the concrete (implemented) one.

#### *Example*

*In the case : « Drawing a house » we have just talked about it, we have seen that we're trying to match the concrete subgoals with the abstract ones but sometimes, it's inevitable to have concrete subgoals which don't match exactly with the abstract ones or which don't exist in the abstract model. The question 1 in the CW method (applied to the kind of subgoals that don't exactly match with the abstract model) will permit, after answered to it, to know if the crossing between the 2 models can be realized easily.*

## **Conclusion**

We can say that the Cognitive Walkthrough can bring a contribution to development of an interface and this contribution occurs at two different places :

First, during the design itself of the interface at the moment where we're making the structure as goals-subgoals of the **projected** task. For example, At this moment, we can use the question 1 : « Will the user try to achieve the right subgoal ? » to know if the concrete subgoals that don't match perfectly with the abstract ones make the distance between them too high. We called it the « reduced CW ».

Secondly, we can put the use of the CW method after that a first design of the interface has been made and correct the interface by a feedback loop to the structure of the **implemented** task in goal-subgoals but at this moment, we have to our disposal the interactive concrete objects of the interface and the actions on their. We can thus apply the entire Cognitive Walkthrough method and not only the subgoals-related questions as it was the case during the design process.

## **3. Cognitive Walkthrough and the design criteria**

### **3.1. Introduction**

Design criteria take the form of a set of basic characteristics leading to an useful and usable interface. These general principles can be used both for the construction of an interface and the validation of it, once elaborated. Indeed, not only these criteria used in the design process of an interface can lead to an useful

and usable interface, but also, when this interface is designed, they can be used to validate the interface by looking if all the design criteria are respected. This is the base principle of another usability inspection method called « Heuristic Evaluation » (see [NIELSEN 93] for more details).

As the experience showed that using these design criteria leads to more useful and usable interface, we should ask ourselves if Cognitive Walkthrough method is taking into account these criteria during the interface analysis. Cognitive Walkthrough doesn't address these heuristics directly. No questions ask whether one criterion is respected or not. But indirectly, Cognitive Walkthrough may support discovery of breached criteria, for instance by looking at the answers given by the analyst at some of the Cognitive Walkthrough questions.

The list of the design criteria is not exhaustive, based on [BODART 98]. For each criterion, the definition will be given and the position of Cognitive Walkthrough according to this criterion will be analyzed. The purpose is not to redefine all the criteria in details, so see [BODART 98] for more explanations about them.

## **3.2. The design criteria and the coverage by Cognitive Walkthrough**

### **3.2.1. Compatibility**

#### **Definition**

A human-computer interface is compatible if the translation of the real world information into system terms is limited.

## Compatibility and CW

As said in the introduction, this criteria is not directly evaluated by Cognitive Walkthrough questions. But, when answering questions 1, 3, 4 and 6, the analyst gets some indications that this criterion is respected or not.

For a given subgoal, the analyst must answer question 1 « Will the user try to achieve the right subgoal ? ». He may answer « No » to this question whereas the system is providing information about which subgoal is the next to achieve, if he thinks that the information provided doesn't match the user's language, for instance. As stated in question 1 explanation, this helps to reveal the gap between the concrete and the abstract task and thus, if the goal/subgoals decomposition of the projected or implemented task (of the interface) corresponds with the abstract one. So, compatibility is the central problem addressed by the question 1.

The « Problem Sheet » questions, to answer when a problem is likely to exist, force the analyst to find the cause of the problem but it is mainly thanks to his own experience that the analyst can find out that there is a problem of compatibility in the interface. Indeed, after giving some negative answers to the Cognitive Walkthrough questions related to compatibility, the analyst knows that there is thus a possible problem but it remains his job to find in which way compatibility is not respected : the method doesn't guide him to locate the source of the problem.

Question 3, 4 and 6 also addresses compatibility as they aim to highlight if users perceive and understand the language of the interface : if they don't, perhaps the effort to translate the interface language into abstract task terms is too important.

To conclude, Cognitive Walkthrough can highlight problems due to lack of interface compatibility but doesn't explicitly tell the analyst the cause of the problems.

### **3.2.2. Consistency**

#### **Definition**

A human-computer interface is consistent if the data and actions are easily recognizable and usable (thanks to their consistency in the interface). The same means must be used to reach the same goals. If users know that the same action will lead to the same results, they will find easier to use the system because they know part of the system that will be always the same even in the unknown parts of it.

#### **Consistency and CW**

Cognitive Walkthrough evaluates the consistency of an interface thanks to questions 1, 3, 4, 5 and 6. Indeed, an usual answer to these questions is « by experience of the system or another system ». This answer addresses clearly the consistency through the applications and in the application itself. So, answering this affirmative answer to these questions permits to know that the interface respects the consistency criterion.

But, in case of negative answer to these questions, we can hardly conclude to a transgression of this rule. As with compatibility, it is the analyst responsibility to find out that some problems are due to a lack of consistency or not.

### **3.2.3. Work load**

#### **Definition**

An human-computer interface is efficient in terms of work load if the amount of data to manipulate and actions to perform for a task is reduced.

Indeed, the more the actions of users, related to a limited number of data, are short, the more the interaction is rapid. The less the user is distracted by foreign information to the task, the more the user is efficient in the achievement of his

task. For instance, if a user doesn't perceive an action, it's perhaps no clear way to use it is available or because the system provides too much information (e.g. we can imagine a professional drawing editor with more than thirty icons giving access to as many actions. The number of icons may prevents users to find the one they need to enable the action required.).

The goal is to minimize the work load in the human faculties scope and to guarantee a performance.

### **Work load and CW**

The work load criterion is barely evaluated by Cognitive Walkthrough. To take again the example given previously, in a professional drawing editor, users may not perceive an icon related to an action because there are too many icons available. So, here it will be quite easy for the analyst to conclude that the user's memory load is not respected. But this case is an exception.

Indeed, most of the time work of load is not evaluated at all. By example, there are no means to evaluate the performance of users in the realization of the considered task. The method is not adapted to this kind of evaluation. If designers want this type of information, they are likely to use other methods like GOMS that offers performance information.

### **3.2.4. Adaptability**

#### **Definition**

A human-computer interface is adaptable if it has the matching faculty in relation with its user. The aim is to provide the user with different ways to perform his task according to different parameters that the user can customize.



## **Adaptability and CW**

As CW is a task-centered method, given a task, it only analyzes the likely usability problems that can be encountered by users. Adaptability is not at all considered by cognitive walkthrough. This method is too action-focused to take this criterion into account.

### **3.2.5. Dialogue control**

#### **Definition**

A human-computer interface is with explicit control if it can give the illusion to be under users' control.

#### **Dialogue control and CW**

This criterion is not evaluated by Cognitive Walkthrough. Neither in the questions nor in the answers, Cognitive Walkthrough doesn't address this problem.

But supposing an interface mostly with explicit control, suddenly the interface triggers off actions. This can confuse users by forcing them to realize actions that are not part of their mental representation of the task, translated into the system terms. This can show an usability problem as users will be lost and confused. In this case, dialogue control is taken into account but only because it showed an usability problem found by the analyst.

### **3.2.6. Representativity**

#### **Definition**

A human-computer interface is representative if used codes, menu items and labels ease the encoding and the memorizing. The aim is to expand the use of significant names within the dialogue.

## **Representativity and CW**

The phenomenon of memorizing or learning is not directly evaluated by the Cognitive Walkthrough. However, the method evaluates the use of significant labels, menu items, shortcuts and all information provided by the interface. For instance, when answering question 1 and 4, i.e. « Will the user perceive the correct action ? » and « Will the user associate the correct action with the subgoal they are trying to achieve ? » , the analyst may answer « Yes » or « No » following that the labels associated to the action are significant or not.

### **3.2.7. Guidance**

#### **Definition**

A human-computer interface is guidance efficient if it informs constantly users about the outcome of their actions and about the position in the realization of the task.

#### **Guidance and CW**

The guidance criterion is one of the most evaluated design criteria by the Cognitive Walkthrough. Indeed, question 1, 3 and 4 evaluates the pre-execution guidance provided by the interface. These questions aim to find out if the system provides good information that lead users to know which subgoal to achieve or which action is the next to perform. Question 5, 6 and 7 evaluate if the interface provides post-execution feedback that informs users about the results of their actions and the progress they have made in the completion of the task.

### **3.2.8. Error management**

#### **Definition**

A human-computer interface has effective error management if it can detect users committed errors and is user-friendly in the way to recover. The goal is to avoid errors as much as possible.

#### **Error managing and CW**

As already said, Cognitive Walkthrough is a task-centered method. So, the analyzed characteristics of an interface depend upon the choice of the task considered. Most of the time, after choosing a task, a task description is elaborated. This task description represents the most common way for the user (as defined in the user profile with supposed background and knowledge) to perform the task. Following this task description, the analyst will go through the questions and see where some problems may occur.

But, if no error-recovery scenario is included explicitly in the task description, it won't be evaluated by the questions of the method. So, the error managing can be evaluated but only in the cases foreseen by the analyst in the task description [FRANZKE 95].

Now, as synthesis, we 'll show you a matrix which relates the different criteria with the questions of the Cognitive Walkthrough method. The aim of this matrix is to show which criterion (criteria) are indicated to reply to certain questions. We have already related the questions of CW above and so, this matrix will be used as a summary.

	Compatibility	Consistency	Work load	Adaptability	Dialogue control	Representativity	Guidance	Error managing
Question 1	X	X				X	X	
Question 2								
Question 3	X	X					X	
Question 4	X	X				X	X	
Question 5		X					X	
Question 6	X	X					X	
Question 7							X	

A matrix like this has an utility. It can help the analyst performing the CW by bringing justification for affirmative and negative answers. For example, if the answer to the question 1 is « No », then it's maybe an error of compatibility with a bad guidance to help the user passing from his abstract task to his concrete one.

These criteria are means to justify answers. It is enough obvious because these criteria are golden rules which have to be respected for any interface.

### 3.3. Design criteria versus Cognitive Walkthrough

#### 3.3.1. Position of the problem

The use of design criteria is required to produce an usable interface. These criteria are golden rules that must be respected during the process of the interface design. But, as already said before, these criteria can also be used as a validation tool. Heuristic Evaluation is an usability inspection method based on similar principles. The analyst has to scan the interface by using all these criteria one by one. As these criteria are general, the problems found with this method are often general ones (i.e. problems of consistency between screens). The weakness of this method is the lack of guidance rules to perform the systematic examination of the criteria and to know which features to look at : the process is unstructured and the only guidance is provided by a list of heuristics. But this free-form nature and lack of a specific process to follow make this technique easy to learn and apply.

Cognitive Walkthrough is a task-centered method. After choosing the task to consider, the method offers a set of questions to apply systematically. Cognitive Walkthrough is better to focus on precise characteristics of the interface but there is a lack of a general view of it [WHARTON et al. 92]. But

*« Cognitive Walkthrough appears to discourage exploration, limiting the evaluator's ability to find problems not directly related to the tasks being performed » [SEARS 97].*

As we described in the previous section, Cognitive Walkthrough sometimes allows evaluation of some criteria but not all, and not in a complete way. But the structure of Cognitive Walkthrough make it bundersome and time-consuming.

So, the two techniques address different fields of usability inspection. The design criteria look at the general features of the interface and Cognitive Walkthrough focuses on the task and the problems in its execution. One method is not better than the other. They just have different objects of analysis. To choose one method

prevents the discovery of problems highlighted by another method. So, a good solution to this dilemma is to perform a method which would combine the advantages of the two methods, neglecting the disadvantage. This method, called « Heuristic Walkthrough » [SEARS 97], is presented in summary in the next section.

### 3.3.2. Heuristic Walkthrough

Heuristic Walkthrough is a new usability inspection method whose goal is to combine benefits of Heuristic Evaluation and Cognitive Walkthrough. The contribution of Heuristic Evaluation consists in a list of heuristics and a free-from evaluation. The failing of HE to focus evaluators' attention on specific features of an interface is filled by the contribution of Cognitive Walkthrough.

Heuristic Walkthrough intends to include both a free-from evaluation and a task-centered evaluation, in a two-pass process. The method comprises a list of user tasks, a list of usability heuristics and a list of questions.

#### Step 1 : Task-centered evaluation

The list of tasks should include not only representative tasks (frequent or important ones) but also less common tasks to have the maximum coverage of the interface. Moreover, each task has a priority that alerts evaluators to the importance of tasks :

*« (...) task priorities should guide evaluators when selecting appropriate tasks to explore ». [SEARS 97]*

While exploring tasks, evaluators should be guided by questions derived from the Cognitive Walkthrough . In [SEARS 97] the following ones are given :

- Will users know what they need to do next ? Is it possible that they simply cannot figure out what to do next ?

- Will users notice that there is a control available that will allow them to accomplish the next part of their task ? It is possible that the action is hidden or that the terminology does not match what users are looking for ?
- Once users find the control, will they know how to use it ?
- If users perform the correct action, will they see that progress is being made toward completing the task ? Does the system provide appropriate feedback ?

## **Step 2 : Free-Form Evaluation**

During the second step, evaluators are not to follow some specific tasks. They can focus on any interface feature they want. However, they will perform an Heuristic Evaluation with a task knowledge gained during the first step. The list of heuristics can be the same as those presented in the section about HE in the first chapter, i.e. :

- Visibility of the system status ;
- Match between the system and the real world ;
- User control and freedom ;
- Consistency and standards ;
- Flexibility and efficiency of use ;
- An esthetic and minimalist design ;
- Error prevention ;
- Recognition rather than recall ;
- Help and documentation ;
- Help users recognize, diagnose, and recover from errors.

## Conclusion

Heuristic Walkthrough will provide more guidance than a list of heuristics, which is useful mainly if evaluators are not so much experienced in usability methods. Further, it appears that the task focus prevents HE from identifying too much false positives. A false positive problem is a problem that is predicted but that will not appear with real users. It also helps as, through task priorities, the evaluators knows about the most important part of the interface to evaluate.

The HE step supports discovery of more general problems than with sole use of CW.

*« Using an appropriate combination of the two approaches allowed evaluators to find numerous less severe problems while avoiding false positives. The key appears to be in balancing the amount of time dedicated to the two types of evaluations » [SEARS 97].*

## 4. Conclusion

We have presented the two different locations of the Cognitive Walkthrough in a design development process, i.e. the task analysis. We have also seen the design criteria. After this, it's time to find out the different outputs provided by the Cognitive Walkthrough and their places in the task analysis. We will discuss about that in the next chapter.



## OUTPUTS OF THE COGNITIVE WALKTHROUGH

### 1. Introduction

This chapter is aimed to present the results of the Cognitive Walkthrough. What are the different outputs of the Cognitive Walkthrough? What are their purposes?

After that, we will try to present a global view of the location of the Cognitive Walkthrough in the task analysis by detailing the different inputs and outputs needed of the different elements of the task analysis and the Cognitive Walkthrough.

### 2. Results of the method and their interpretation

After performing the Cognitive Walkthrough method, i.e. after answering the questions for all the subgoals and actions, the analyst has a list of all the questions with their answer related to all the subgoals and the actions of the selected task. When a negative answer is given, there may be a problem in the interface. By negative answer, we mean that the answer is either a categorical « No » or « Not always » which is equivalent to « Perhaps ». So, the negative answers include all the answers except categorically positive ones.

For each question where a problem may occur (i.e. each question where the answer is no affirmative, i.e. « No » or « Perhaps »), a problem record sheet must be fill in. This sheet includes the number of the problem (in order they are

found), the description of the problem, the location in the execution of the task (the interaction point), the likely concrete difficulties encountered by users in context, the assumed caused of the difficulties, the question to know if the question leads or not to a failure case [LAVERY, COCKTON 97], the seriousness of the problem [JOHN, MASHYNA 97], the concrete interactive object (CIO) of the interface concerned and a design suggestion to solve the problem. We will now explain you these notions :

## **2.1. The problem number**

Each problem must be numbered in the order it is found.

## **2.2. The description of the problem**

The evaluator gives a brief description of the problem found.

## **2.3. The interaction point**

The interaction point is defined by the scenario number, the path to the subgoal and/or action where the problem was found and the number of the CW question where a negative answer was given (e.g.: subgoal A, action 3, CW5 or subgoal 1.1.2, action 1, CW3 or subgoal 2.1., CW 1).

## **2.4. Likely concrete user difficulties in context**

Explain here what will result of the usability problem : will the user make a wrong action? Will he gives up the task or restart in another way? Etc. Justify this in the context : try to say which action the user will perform, what will be the concrete result,...

This will let the analyst see the consequence of the problem. He will be able to compare this consequence with the one once the problem will be resolved.

## **2.5. The assumed causes of these difficulties**

The causes are the justifications brought to answers of questions performing during the Cognitive Walkthrough (see chapter 3).

## **2.6. The success/failure case**

After answering all the questions during the Cognitive Walkthrough, one can then consider all which have been answered in a negative way. Now, with the list of negative answers and their related questions, it's important to find out the failure case. A failure case is related to one kind of user, on one specific task and in one specific context of work. It's the case where the user can't achieve his task. It prevents him from achieving his task in a efficient way. By efficiency, we mean that the user will achieve his task without being jammed in a place which prevents him to finish his task.

One question where we have answered in a negative way doesn't immediately lead to a failure case. The problem the analyst will meet is to detect a failure case

simply by looking answers (and especially their justification) from questions replied in a negative way. We consider each question in detail below.

### 1.1. First question: "Will the user try to achieve the right subgoal?"

If we answer negatively at this question, it's probably because this subgoal is too far of the user's abstract task as mentioned in [Clayton, Rieman] and we face a failure case.

#### *Example*

*An user with only web-experience doesn't know that he has to select an object before to manipulate it, by example to enlarge it. So, if we have the following scheme (Fig.19):*

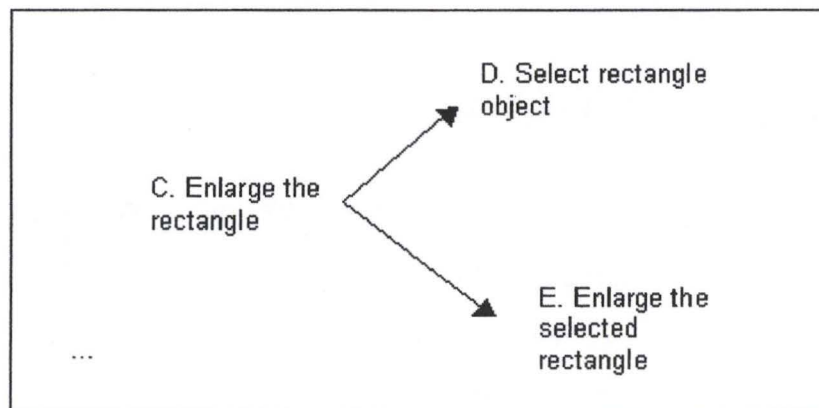


Figure 19 - Goal/Subgoals decomposition of the subtask "Enlarging the rectangle"

*The user will probably answer: "No" for the first question during the analysis of subgoal D because on the Web, he hasn't to do this kind of selection. We can say that he has never done this kind of selection on the Web. However, we can say there is a problem with the Web user and this interface, so there is maybe a failure case for this*

*specific cognitive walkthrough (a specific user, a specific context of work, a specific task, a task description, a cognitive walkthrough analysis).*

*However, the system, via the interface, should have put a prompt to tell him the next step to achieve and how to achieve it, assuming the user profile and the context of work.*

*So, this is important to specify the right assumptions in the part: "User profile". If we had took a Windows user, the answer at this question would probably be "Yes, by experience of the Windows system".*

Finally, let's notice that more the subgoal is a high-level subgoal, more the failure case will be important because the user will be jammed before to try to perform all the subgoals of the considered one.

- 1.2. Second question: "What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?"

If we answer negatively at this question, this probably leads to a failure case, at least for the specific kind of user, the specific context of work and the specific task we have selected for this specific cognitive walkthrough. The failure case is more important if the subgoal where we have answered negatively at this second question is a high-level subgoal.

- 1.3. Third question : Can the user perceive that the correct action is available ?

If the user can't perceive the action, he won't achieve it, so, this surely leads to a failure case.

- 1.4. Fourth question : Will the user associate the correct action with the subgoal they are trying to achieve ?

If the user faces a problem here, this doesn't lead necessarily to a failure case because he has however performed the action but can say to himself that this is not the good action because he doesn't associate it with the subgoal he is trying to achieve and so can cancel the action and arrives to a failure case.

1.5. Fifth question : Will the user perceive the feedback ?

If the user doesn't perceive any feedback, this perhaps leads him to a failure case. This is for the judgement of the analyst.

1.6. Sixth question : Will the user understand the feedback ?

If the user doesn't understand the feedback, this perhaps leads him to a failure case. This is for the judgement of the analyst.

1.7. Seventh question : Will the user see that progress is being made towards solution in relation to their main goal and the current subgoals ?

If the user doesn't see any progress, this perhaps leads him to a failure case. This is for the judgement of the analyst.

To conclude this sixth point, we can say that one question (from 3 to 7) where we answer "No" doesn't lead automatically to a failure case. We have to consider the number of "No-answered" questions (from 3 to 7), their frequency and their gravity (as mentioned in [JOHN, PACKER 95]), the analyst must rely on his own judgement here. For the question 1 and 2, a negative answer is more a sign for a failure case. However, the analyst has again the last word to decide if it is the case or not.

One failure case is a proof that there is a problem in **one** task of the entire tasks supported by the interface for **one** specific user and for **one** specific context of work.

## 2.7. The seriousness of the problem

1. While the section « assumed causes of the difficulties » forces evaluator to exactly locate usability problems and their causes, determining the « seriousness of a problem » permits the evaluator to put in evidence which problems are critical for performing the task and which ones that are light.

The most important problems can be isolated both by looking at the utility and usability criteria (Chapter 1, point 2.) we have defined sooner in the task analysis and by the aspect of success/failure case we have defined in the previous point in this chapter. The aim is to prioritize them in the resolving process.

Thus, A negative answer has to be evaluated in regards with the balance of the utility and usability criteria we have to our disposal and in regards with the success/failure case.

The success/failure case has already be took in consideration in the previous point in this chapter. The conclusion is that a failure case leads to a serious problem.

This point aims to see the influence of the utility and usability criteria in the seriousness of the problems.

The balance of the utility and usability criteria which are inferred, among others, from the task analysis doesn't have an influence on the manner to

perform the CW, i.e. answering questions (with justifications) of the CW . They « only » have an influence on the manner to interpret the results according to the balance allowed on their. By « interpret the results », we mean categorize each problem, i.e. put a level of seriousness on it.

In other words, the act of replying in the same way to the same question during the accomplishment of the CW can, according to the balance of the utility and usability criteria, may result in two different interpretations of the interface. We will give you an example to reinforce this explanation.

#### *Example*

*Let's consider two distinct interfaces.*

*The first one has revealed the most significant utility and usability criterion :*

- *An average rapidity of execution*

*The second one has revealed the most important utility and usability criterion :*

- *Error rate to zero*

*Consider now the question 3 in the CW method : « Can the user perceive that the correct action is available ? » and let's assume that in the two interfaces, the answer is « Perhaps » because the interface provides two likeness actions leading to an ambiguity (but in the two interfaces, only one action is correct).*

*Thus, we notice that in the two cases, the answer is about the same and this possible mistake will perhaps delay the achievement of the task.*

*Taking in consideration the seriousness of this problem according to the utility and usability criteria, the first interface only encounters a light-to-average problem*



*(according to the rapidity of execution which may be average) while the second one encounters a big problem as the error rate has to be zero.*

So, we can say the utility and usability criteria haven't an impact on the manner performing the CW, but have a big impact on the manner interpreting the results of the CW.

2. We have tried to draw a matrix (Fig. 21) which reveals, for any of the answers of the questions of the CW method, the seriousness of a problem in the interface according to the balance of each of the utility and usability criteria.

		Learning time			Rapidity of execution			Error rate			Period of persistency			Subjective satisfaction			Degree of coverage		
		low	aver.	high	low	aver.	high	low	aver.	high	low	aver.	high	low	aver.	high	low	aver.	high
Each question of the CW	YES																		
	PERHAPS						x	x					x			x			
	NO	X	x				x	X	X	x			x	X		x	X		

Figure 21 - Matrix of the seriousness of the problem according to the utility and usability criteria

Legend :

X : Serious problem

x : Problem

This matrix supports interpretation of the results of the CW, i.e. the answers to the questions, according to the balance of each of the utility and usability criteria in a way of putting in evidence the big and the little problem(s) in the interface.

*Example*

*Let's review the previous example and apply it into the matrix.*

*For the first interface, for which we have retained an average rapidity of execution, a negative answer to any of the question leads to a problem.*

*For the second interface, for which we have retained a error rate to zero (so a low error rate in the matrix), a negative answer to any of the question leads to a serious problem.*

**Remark**

This matrix seems to be very simple, as each negative answer added to a certain balance of a criterion, will lead to a problem or a serious problem.

Let's notice the results brought by this matrix has to be qualified according to the current evaluation and must not to be take into account in the absolute, out of context.

In fact, we remind the evaluator has to assess the seriousness of each problem according both to the success/failure case and to the utility and usability criteria. This matrix will permit to provide him some guidance for the assessment of the seriousness according to the utility and usability criteria.

## **2.7. Concrete interactive object concerned by the problem**

This is simply the mechanism of the interface concerned by the problem (e.g. a menu, an icon, an information feedback, ...)

## **2.8. A design suggestion to solve the problem**

The evaluator can try to suggest a correction of the design where the problem occurs. Two cases can be occur.

First, the CW has been performed during the task analysis and it's enough easy to change the prototype.

Secondly, the CW has been performed after the interface or prototype was made completely and we have to make a feedback to the step of the structure of the task where the decomposition in goal-subgoals of the projected task is made.

## **3. Conclusion**

We see now how the different outputs of the Cognitive Walkthrough have to be inserted in a the task analysis walkthrough that we have choosen. To this purpose, we present a scheme (Fig.22).

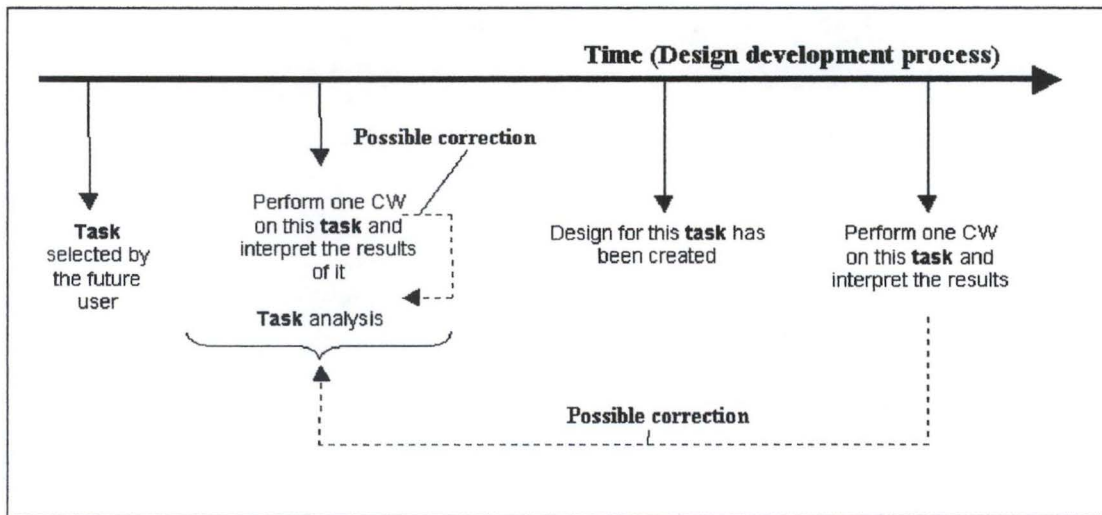


Figure 22 - Interface design evolution with Cognitive Walkthrough

One task, selected by the user, has to be implemented (in an interface). It can be any task which can be computerized. The analyst will perform the task analysis on it, as it is described before. During the task analysis, a CW method and the interpretation of these results can be performed as well as possible, in regards with the poor material for the future interface. Indeed, even if the structure in goal-subgoals of the projected task already imposes some features and mechanisms of the future interface, we haven't maybe not yet an interface and maybe even not yet at least a prototype. But applying the CW here can maybe warn the analyst and permit him to resolve certain problems. That can involve some corrections to the structure of the projected task. Some iterations may occur.

The application of a such CW only applies to subgoals, so to the question 1 and 2. Indeed, for the moment, we have only the decomposition in goal/subgoals. This is called a reduced-Cognitive Walkthrough

After that, a design is made. Then we will perform again a CW and the interpretation of these results on this specific task using the concrete (implemented) goal structure and possibly a correction to the task analysis in the structure of the projected task (Fig.22).

The following scheme (Fig.23) is more accurate, putting in evidence the place of the CW in the task analysis and after it. This scheme will permit to see which inputs are needed to the CW and explained where the analyst can find their in the task analysis.

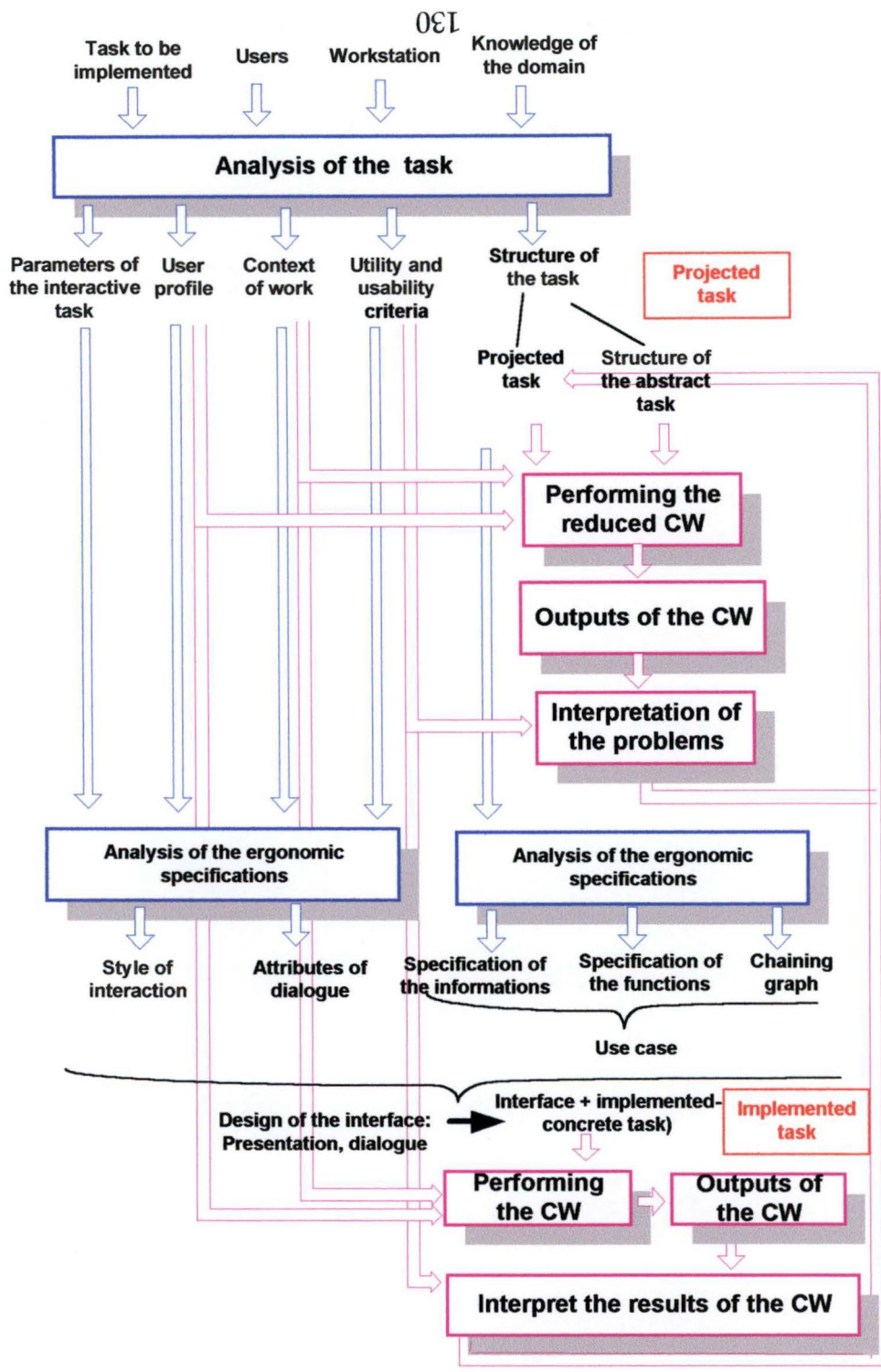


Figure 23 - Cognitive Walkthrough in the design process

# PART 3



DESIGN OF A TOOL SUPPORTING THE COGNITIVE WALKTHROUGH  
VIA THE TASK ANALYSIS WALKTHROUGH

## **1. Introduction**

Performing the Cognitive Walkthrough method takes a lot of times. The time is spent between the task description to realize, the answering process of the questions for each subgoal and action. The method produces thus many papers written by the analyst.

So, we'd like to develop a software that will make the performing of this method easier, less time-consuming and bundersome. For the task description, the analyst must be asked the goal/subgoals decomposition in a clear and easy way close to how he would have done it by himself. Moreover it must obviously be permitted to step back if an error has been made and that the analyst wants to correct it. At this point, the system has to guide the analyst in this process.

After this first phase is over, the analyst must be able to perform the method itself. That means that he has to answer the subgoal-related questions and the action-related ones. Once again the guidance provided by the system must be maximum. In the order of subgoals, the system must ask the analyst to answer the questions. And for atomic subgoals, to answer the questions related to

actions<sup>4</sup>. If the analyst, at some point, detects a usability problem, the system must provide help to record it.

Then, after performing the method, the problems found must be accessible for consultation.

Moreover, as to record the actions, a list of UAN actions will be proposed to the analyst, still in the point of view of making the method faster, the possibility is left to the user to custom the set of UAN actions available by creating his own UAN libraries.

The specification of this Cognitive Walkthrough editor will be built from a task analysis walkthrough as developed by Professor Bodart in [BODART 98]. A projected task will be so defined. Then, we will specify the ergonomical and functional specifications. From these, we will be able to build a prototype of interface. The last part of this chapter will present a synthesis of the results obtained by performing Cognitive Walkthrough on the projected task of this editor, the modification suggested, and the results of an evaluation on the implemented task, still with CW.

## **2. Context Analysis**

### **2.1. User Profile description**

The population of users includes two different kinds of possible users :

The first likely user is a Cognitive Walkthrough expert. He is used to all the steps of the method, to the way to answer questions and to the description of the task and the use of the UAN language.

---

<sup>4</sup> This has been detailed in the third chapter.

- **Task experience** : good ; as being expert, they know how to perform the method.
- **System experience** : good, as being a usability expert, testing system interfaces, supposing that he is used to the Windows environment is not a heavy assumption.
- **Motivation** : high, because they are expert as well of the task as of the system. They will surely be productive.
- **Experience of a complex interaction technique** : high, because their ability to manipulate the keyboard and the mouse is important.

The second possible kind of user is a software designer or usability evaluator, but without much Cognitive Walkthrough experience. They just know the basic principles of the method : the need for a task description of the task to analyse and the process of stepping through the different questions for each subgoal and action.

- **Task experience** : elementary, because they just know about the prescribed task as it is described in the Cognitive Walkthrough papers. However, we must assume that they have never performed a CW on paper.
- **System experience** : good, as being software designer or usability evaluators, they are very likely to be used to the Windows environment.
- **Motivation** : good. They know a bit about the task and are used to the system, but they will somewhat be in a learning process during which they will first discover the system.
- **Experience of a complex interaction technique** : high, because their ability to manipulate the keyboard and the mouse is important.

## **2.2. Analysis of the workstation**

### **2.2.1. Physical environment**

The physical environment is the common environment of any office work. It includes a personal computer with the common interaction technique such as keyboard and mouse. The condition of work are likely to be the same as in any informatic project : delays to respect, team work,... It is not probable to lead to stress situation. Thus, we can assume that the system will be used in normal condition of the environment and of the user.

### **2.2.2. Tasks allocation**

- **Person** : as explained in the user profile, the people who will perform the task are either the usability expert or a designer involved in the design process of the software to evaluate with the Cognitive Walkthrough.
- **Function and role** : any of the two possible users are responsible for any of the tasks intervening in the Cognitive Walkthrough. However the task of creating the task description should be ideally done by the task analysis. The evaluator must only insert the task structure into the system.

### **2.2.3. Treatment type**

The treatment type is here multi-task. Indeed, the evaluator can on one side perform the different step of the Cognitive Walkthrough as proposed by the system and on the other side, at each step, switch to the prototype or interface mock-up that he is evaluating. This to permit him to give the system the information it requires.

#### 2.2.4. Task execution modalities

As performing the CW takes some time, users must be able to interrupt any task involved in the method and to go back afterwards. For one instance of the performing of the method, users are of course not allowed to perform two different tasks at the same time. It would be crazy to do it !

### 2.3. Descriptive parameters of the tasks

- **Prerequisite** : average, the prerequisite required are only the ability to use the different interactive elements and mechanisms (click, drag&drop,...) of the Windows environment. But users must know the basis of the prescribed task of performing a Cognitive Walkthrough.
- **Productivity** : if the CW is performed by a usability expert who only makes evaluation of interface of different systems, the productivity is high, because he will use the system each time he has to perform an evaluation. But if it a designer's job to evaluate the interface, the productivity will be low because he will only use the system to evaluate the software he is working on. Then he won't use the system again before his next project.
- **Objective environment of the task** : users can manipulate external documents resulting from the task analysis such as user profile of the system to evaluate, the task description of the implemented task, the context of work, the utility and usability criteria and the task description of the real task. The interface mock-up or prototype is also part of the environment.
- **Environment reproducibility** : partly reproducible. The task description of the implemented task should be described by the system. Idem for the utility and usability criteria. Anyway, all the concepts of the Cognitive Walkthrough may be reproducible. However, we will not reproduce the abstract task and

the context of work as it seems not valuable to implement. These documents will still be used by evaluator as external to the editor.

- **Task structure** : the degree of freedom in performing the tasks is rather low seeing that the guidance provided by the system must be high to accelerate the performing of the method. Further, the order in which to perform the different tasks of the editor must be respected. However, as said in the task execution modalities, users must be allowed to go back in the task, permitting them to change some of the information he gave to the system. Moreover, the opportunity is let to users to interrupt a task and to take it back later (by saving/loading).
- **Task importance** : average. The evaluation of the interface of a system to design is an inevitable step of any design process. So, performing the task of an evaluation is a crucial task to ensure the success of a system. But as there are many evaluation methods, the task of performing the CW can be replaced by any other task of evaluation with another method.
- **Task complexity** : high, because performing the CW is a complex task that requires much reflexion from the evaluator. But there is no complexity attached to the necessary manipulation of the method.

### **3. Tasks structure**

#### **3.1. First task : Enter the inputs of the CW**

##### **Decomposition into goal/subgoals**

1. Enter the inputs of the CW
  - 1.1. Give information about user profile
    - 1.1.1. Enter user system knowledge
    - 1.1.2. Enter user system knowledge description
    - 1.1.3. Enter user task knowledge
    - 1.1.4. Enter user task knowledge description
  - 1.2. Give information about the tool to evaluate
    - 1.2.1. Enter the tool name
    - 1.2.2. Enter the version of the interface
    - 1.2.3. Enter the utility and usability criteria
  - 1.3. Give the description of the task considered
    - 1.3.1. Give a text description of the task
    - 1.3.2. Create the goal/subgoals decomposition
      - 1.3.2.1. Enter the goal name and description
      - 1.3.2.2. If it's an atomic subgoal, specify the sequence of actions

Plan 1.3.2.: do [1.3.2.1.-1.3.2.2.] until the task description is completed.

##### **Diagram of goal/subgoals**

See Fig.24

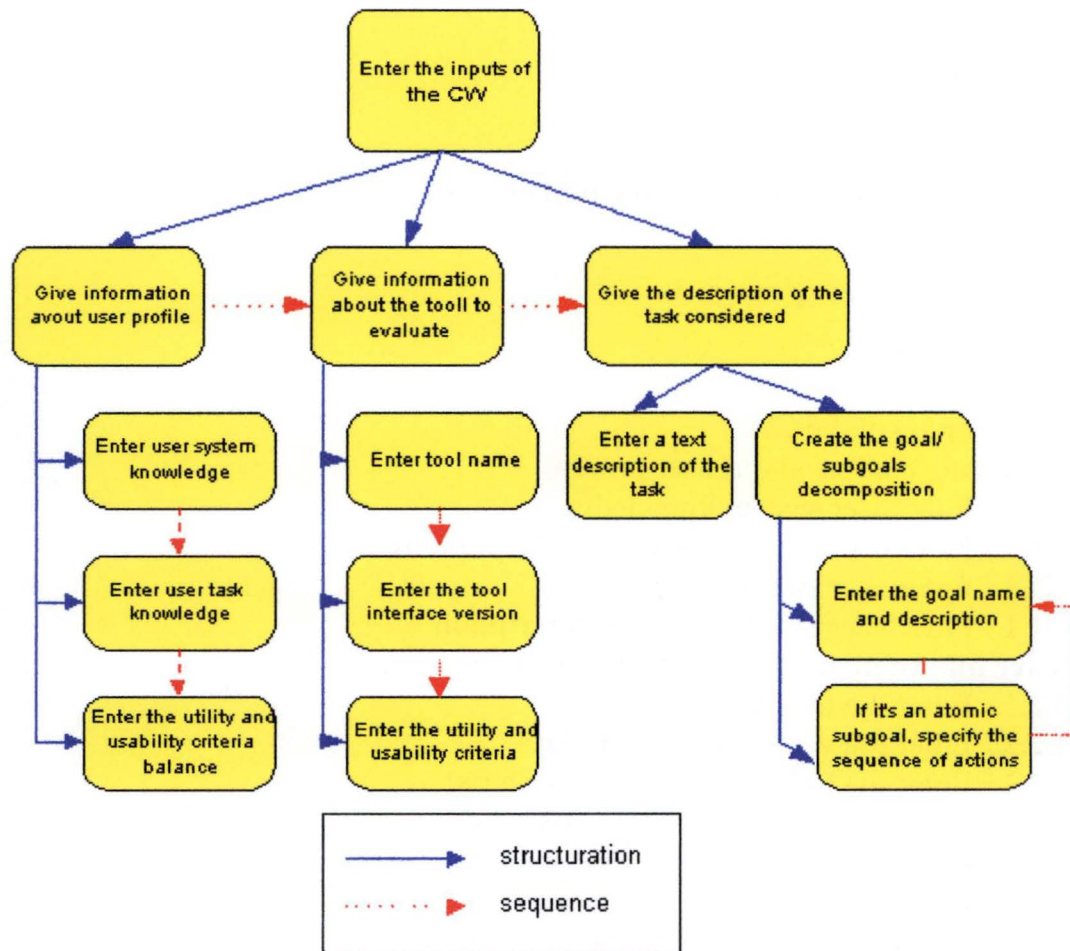


Figure 24 - Goal/Subgoals decomposition of the task "Enter the inputs of the CW"

## 3.2. Second task : Performing the CW

### Decomposition into goal/subgoals

#### 2. Perform the CW method

##### 2.1. Answer questions for one subgoal

##### 2.1.1. Answer the subgoal-related questions for a subgoal

2.1.1.1. For each question, choose usual answers or customize one

2.1.1.2. If a problem is found, record it.



- 2.1.1.2.1. Give problem description
- 2.1.1.2.2. Give the likely difficulties caused by this problem
- 2.1.1.2.3. Specify CIO concerned
- 2.1.1.2.4. Specify the assumed causes of the problem
- 2.1.1.2.5. Specify the seriousness of the problem
- 2.1.1.2.6. Specify a design solution to this problem
- 2.1.2. If the subgoal was atomic, answer the action-related questions
  - 2.1.2.1. For each question, choose usual answers or customize one
  - 2.1.2.2. If a problem is found, record it.
    - 2.1.2.2.1. Give problem description
    - 2.1.2.2.2. Give the likely difficulties caused by this problem
    - 2.1.2.2.3. Specify CIO concerned
    - 2.1.2.2.4. Specify the assumed causes of the problem
    - 2.1.2.2.5. Specify the seriousness of the problem
    - 2.1.2.2.6. Specify a design solution to this problem

Plan 2.1. : do [2.1.1.-2.1.2.] for each subgoal of the task description.

#### **Diagram of goal/subgoals**

See Fig.25.

### **3.3. Third task : Consult the problem list**

This task is obvious. It only consists in consulting the list of problems found after the two first tasks are performed. Users have only to click to read the problems description. That is why we do not propose here a task structure.

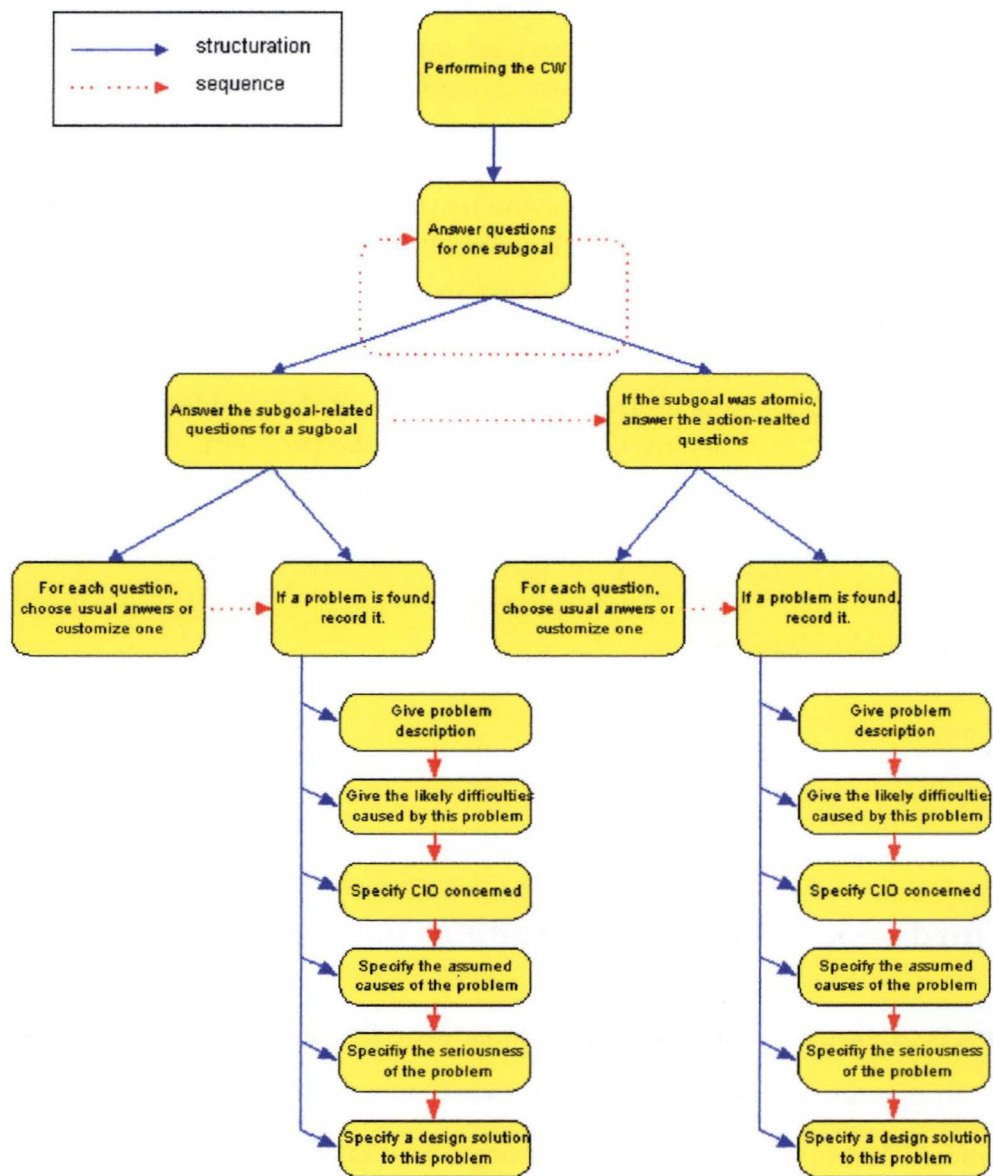


Figure 25 - Goal/Subgoals decomposition of "Performing the CW"

### 3.4. Fourth task : Use new UAN library

#### Decomposition into goal/subgoals

- 4. Use new UAN library
  - 4.1. Create new library
    - 4.1.1. Specify name and location of the library
    - 4.1.2. Create the list of UAN actions
      - 4.1.2.1. Add UAN action
      - 4.1.2.2. Remove UAN action
    - 4.1.3. Search for library
  - Select UAN Library

#### Diagram of goal/subgoals

See Fig. 26.

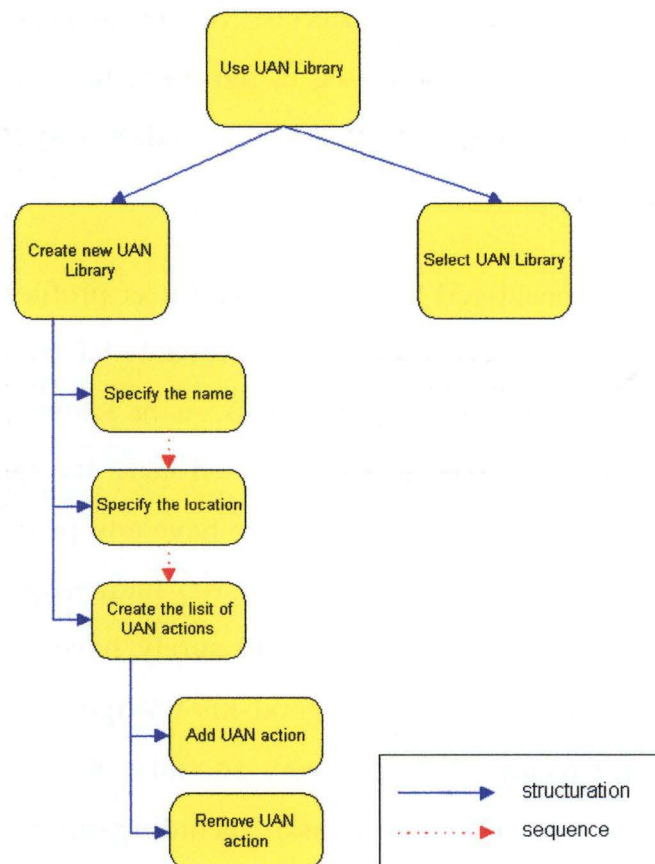


Figure 26 - Goal/subgoals decomposition of "Use UAN Library"

### **3.5. Bootstrapping : Applying CW on the projected task**

As we explained in the fifth chapter, Cognitive Walkthrough can be performed at two different locations in the global process of an interface design. Firstly, after the task analysis when a projected task is defined (the reduced Cognitive Walkthrough). Then, on the implemented task (the standard Cognitive Walkthrough as explained in the third Chapter). We applied CW on the projected tasks. This section aims to describe a synthesis of the results got.

We defined the structure of the projected tasks with the TKS model in the section 3 of the present chapter. Using this goal/subgoals decomposition, we used the reduced CW, i.e. the Cognitive Walkthrough but only taking into account the subgoal-related questions. This aims to detect gaps between the abstract goals structure, as thought by users, and the goals structure of the projected task. If this gap is not evaluated here, it will find itself in the concrete goal structure of the implemented task. So, detecting early this problem allows to correct problems before developing the complete interface.

For each task, we have considered the two different user profiles we defined for the task analysis. The experienced user will not probably have any problem with the goals structure of the different projected tasks. As he knows perfectly how to perform a CW and we transcribed in the projected tasks the same steps, in the same order required for a CW, he is not likely to have any problem. The second point is to know if this user will have the required knowledge to perform the different tasks. In the same way, the user will surely have the knowledge to achieve the different tasks thanks to his good knowledge of the abstract task. However, that doesn't mean that the system should not provide any further information to ease the realization of the tasks. Including information to provide the user information about the progress of the tasks is important, even if not vital for this first kind of user.

Consider now the second kind of users that may use our editor. This user knows about the basic principles of CW but has perhaps never performed one. In this case, the application of the reduced CW has shown that this user won't probably try to achieve the different goals of the different tasks correctly. He might skip subgoals, perform some in another order, and so on... So, he won't have the knowledge of the task structure of the projected task. Moreover, he may be not familiar to some inputs required. For instance, will he know what are the design criteria and how to balance them ? We have to keep this in mind for the following steps of the interface design : the system should provide these information and appropriate help should be available.

In conclusion, the reduced CW has shown the necessity to provide guidance in the editor. The different kind of guidance to be offered are detailed in the section 4 of this chapter.

### **3.6. Sequence of the different tasks**

The three first tasks must be accomplished in a fixed order. Firstly, the task « Enter the inputs of the CW », then « Performing the CW » and finally, « Consult the problems list ». That is so because they are part of the main activity of doing an usability evaluation. The task « Use UAN Library » can be performed from the main screen when users want to use special UAN library or to create new ones.

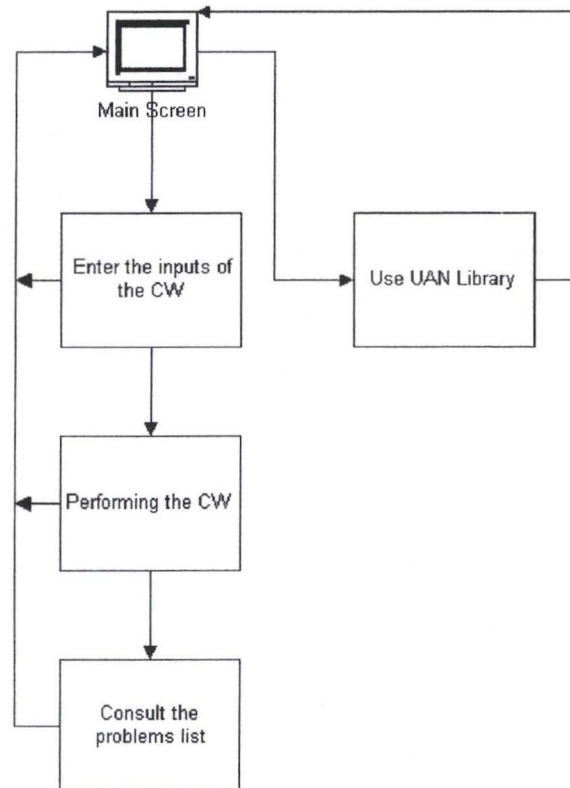


Figure 27 - Sequence of the different tasks

## 4. Guidance

The guidance is a service provided by the interface to users in the process of performing tasks. Two types of guidance are to be distinguished :

- **The active services :** these can be defined as services provided by the interface before the execution of actions by users. They constitute in a way a pre-execution help.
- **The control services :** this second kind of services are more what we call post-execution services, i.e. services that only occur after the execution of an

actions by users. They « control » what was done by users of the system. For instance, the validation of the inputs entered by user.

The abstract task that we want to implement, the Cognitive Walkthrough performing, is a strongly step by step guided task. As the results of the reduced CW suggested it, we would like to provide the same form-based method once implemented with a guidance through the different steps that would be provided to evaluators. So, as in most systems, a guidance of control will be provided to control the information received by the system and so, to prevent a post-execution control of the users' actions. Moreover, several active services of guidance are also desirable to be included.

We distinguish three different important functions of services guidance in the editor we want to build.

#### **4.1. Guidance of context inputs**

The first guidance to provide concerns the capture of the context inputs of the Cognitive Walkthrough : the selection of the utility and usability criteria, the description of the context of work and the profile of the final user of the CW editor. The next figure is related to the data that this guidance concerns.

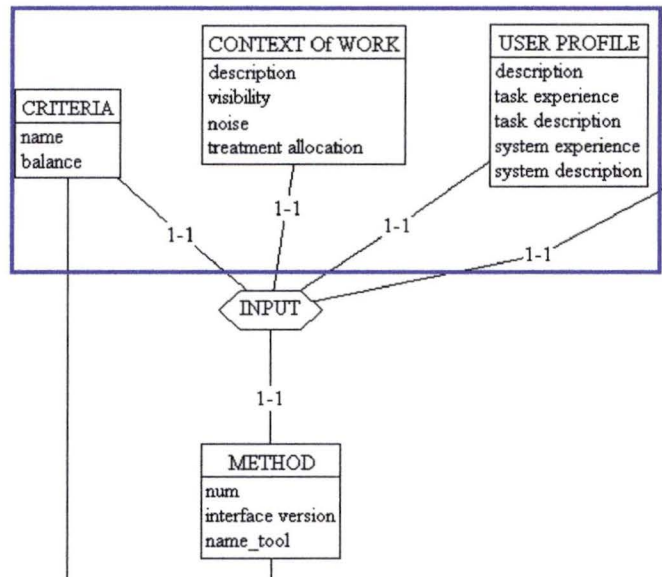


Figure 28 - The data related to the guidance of context inputs

The guidance will be materialized by a kind of *wizard*, similar to those available in software such as Microsoft Publisher. So, users will be brought through the different steps of capturing, firstly the user profile, secondly the context of work and finally the utility and usability criteria associated with the editor. Of course, the wizard must permit to users to go back to make changes in the previous steps.

## 4.2. Guidance of the tree construction

This second guidance service to provide is about the tree construction of the task description, i.e. the goal/subgoals decomposition and the seizure of the actions sequence of the atomic subgoals.



Users have to enter for each goal, its name, description and actions sequence if required. The guidance must include navigation through the different subgoals. In parallel, the guidance must provide, once the goals information capture is completed, the graphical view of the tree in the form proposed by [DIX et al. 93], whom we also talked about in chapter 3.

The following figure (fig. 29) shows the different information of the IS concerned by this second guidance.

The guidance will ask users to enter information in the same order as a task description would be performed on paper : level by level. So, firstly, name, description of the main goal will be asked. Then, the same will be done with any of the subgoals-children. For the subgoals which are atomic, the system must ask the actions sequence. After all the concerning information for the subgoals of one level have been entered, and if not all the current level subgoals are atomic, it is the turn to the next decomposition level to be described. And so on until all the subgoals of a level are atomic.

Once again, the opportunity to step back must be let to users if they want to modify any information.

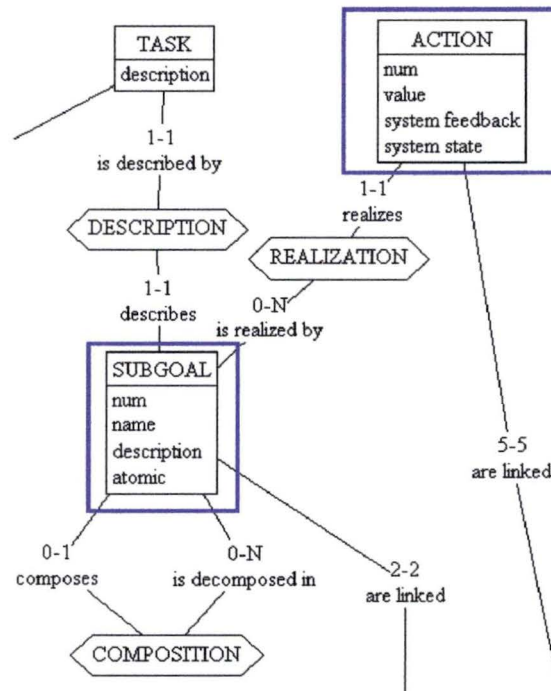


Figure 29 - The data related to the goal/subgoals decomposition

### 4.3. Guidance in answering the CW questions

After the the two first steps, when all the inputs of the method are available, the system must begin to ask users the CW questions in the order required by the method, i.e. from the main goal to the lowest subgoal. Each sugoal and action must be answered in the correct order. The order of answering the questions is described in the fourth chapter. The figure 30 presents the different information of the information system intervening at this step.

Another pre-execution guidance that we must provide is the list of usual answers. According to the question, and the information provided as inputs (user

profile and context), the several « most frequently used answers »<sup>5</sup> must be shows to users to permits them to make envnetually a quick choice between them.

Moreover, when users want to record a problem that they think to have found, a problem record form is to be displayed and permits the recording.

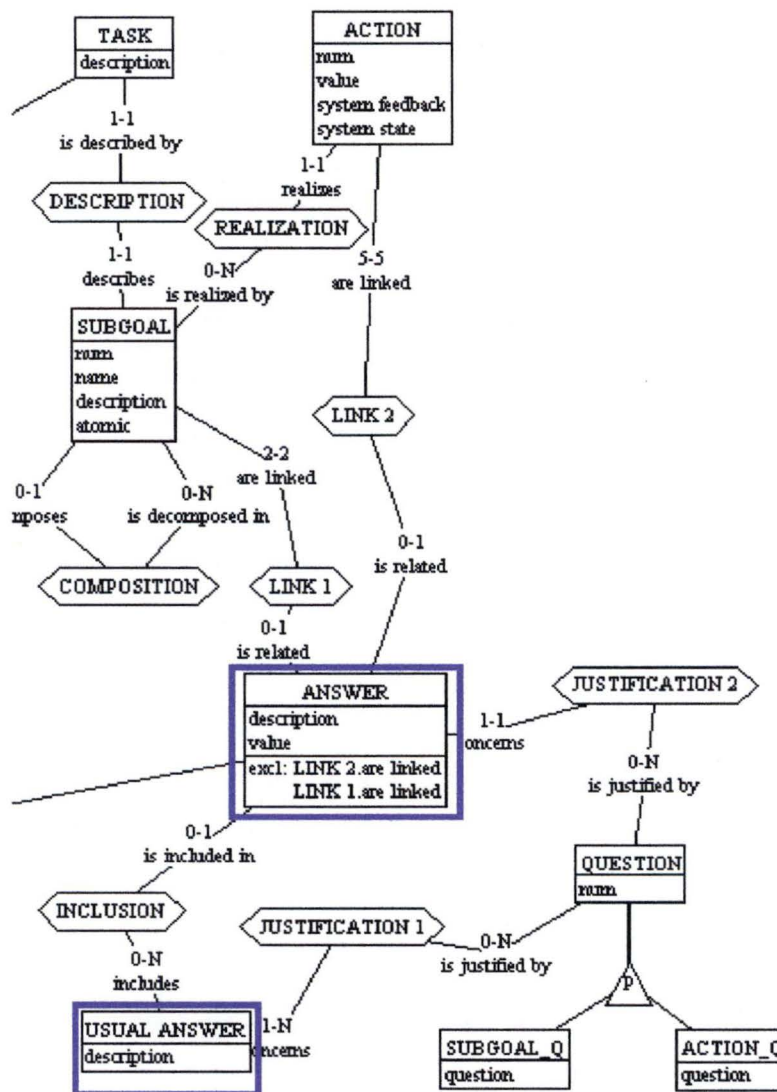


Figure 30 - The data related to the answering of questions

<sup>5</sup> See chapter 3, section 3.2.2.

#### **4.4. General guidance of navigation**

A general guidance of navigation must also be implemented as the order to perform the three steps described above is fixed. Thus, the system must bring users through these steps.

#### **4.5. Help while performing the method**

We asked ourselves about the opportunity to include theoretic help in the different screens we will build. That means, should we give text rephrasing the questions, explaining more about the theory behind the questions (the position in relation with the Norman model of the task), and so on. It seems valuable for the novice user who is beginner either with the editor or the Cognitive Walkthrough method in itself. But for the expert user, which is the second kind of user we have considered in the stereotype of user, all these are extra information and are really not required.

A compromise between these two approaches would be to offer the possibility to call for help containing the theoretical explanation of the method. But we think this help has to be visible during performing the task. We think we to include a help companion similar to those provided in Microsoft Word (fig. 31). It would give information at the current step, that the user is trying to realise, of the task.

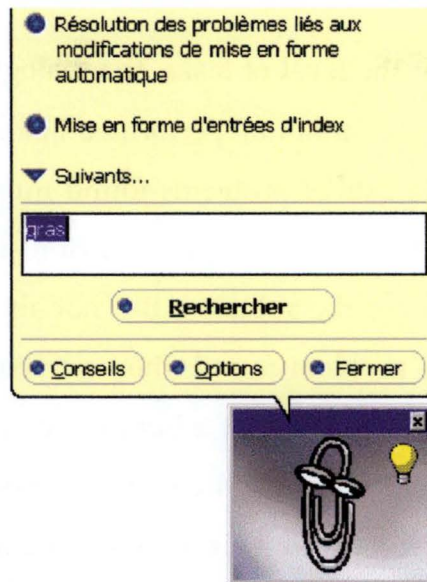


Figure 31 - The help wizard of Microsoft

## 5. Derivation of the ergonomical specifications

### 5.1. Choice of the dialogue attributes

There are four attributes specifying the dialogue of the interface :

1. **The control of the dialogue** : at the global level, the control of dialogue will be internal. As for the real task of performing the Cognitive Walkthrough, users must be guided through the different steps of the method. In the same way as in other softwares, it seems to be appropriate here to use some kinds of *wizards* to guide users. This way of proceeding permits to accelerate the dialogue as, each time it is the system that will ask the information needed and not users that trigger off the different step, and it will reduce the likely errors performed. For instance, if users won't try to answer the CW questions before creating the task description if the system ask them explicitly to insert it before starting the step of answering questions.

2. **The dialogue mode** : at the level of tasks, the dialogue mode is synchronous as one step of the method cannot be performed before the following one. For example, consulting the lists of problems found must not be possible before performing the method in itself. Quite obvious ! Very often, the same reasoning can be made for the subtasks. But not always. For instance, when users want to define a problem found, they are likely to do it in the order specified in the task « Performing CW » but they are not obliged. For expert users, a *wizard-free* version could have been considered but as the different steps of the method and their order are mandatory, no benefit would have have been got.
- However, as the oppportunity to step back in the process, for instance to correct mistakes must be let to users, the dialogue can be partly asynchronous.
3. **The mode of functions release** : it is manual, explicit and displayed because the different functions are triggerd off by users thanks to the actions provided by the system. These actions will be explicitly present, for instance in the form of buttons (OK, Cancel,...).
4. **Metaphor** : it is based on the mini-world. The different forms used in the paper version of the method must be reproduced electronically. Users must be able to fill in the form in the same way as they would have done it on paper. This agrees with what was said about the existing environment and its reproductibility in the descriptive parameters of the task.

Dialogue control	Internal
Dialogue mode	Mixed
Mode of functions release	Manual,explicit, displayed
Metaphor	Mini-world

## 5.2. Utility and usability criteria

The utility and usability criteria are fundamental: they fix the general constraints in term of quality and efficiency of the interface. They constitute the basic principles that the interface must respect at all cost.

- **Learning time** : this criterion seems not to be a crucial criteria for our task. Users can try some times the tasks before performing them for real work. However, a good learning time will obviously ease the use of the system we want to build. As the complete task of evaluating an interface with the Cognitive Walkthrough is time-consuming, an interface with a small learning time with permits to gain time at the very first use of the interface. Moreover, as we want to provide good and strong guidance for evaluator, it is likely that the learning time will be low.
- **Rapidity of execution** : as said before, performing the Cognitive Walkthrough on paper is a bundersome and time-consuming method. This is one reason why an editor would be necessary : to shorten the time required to perform the method by an evaluator, whose time is expensive and rare. So, the rapidity of execution must not be as high as possible but it must be better than the time required by the conventional way of performing the method.
- **Error rate** : by experience of the real task, we know that evaluators often tend to step back and change their answers to some questions. So, the error rate must not be necessarily low. However, error not caused by changes of opinion of evaluators but by, for example, misunderstanding of system messages, must be low because as performing the method takes a long time, recurrent errors may really hamper to perform the task efficiently, and worst, can lead evaluators to be fed up with the task and give it up.

- **Period of remanence** : this criterion is quite important as, if the learning time may not be always the shorter as possible, it is important that users recall how to perform a task from one use to another. For usability expert which frequently use the CW method to evaluate interface, the period of remanence may be low as the time between two different use of the tool will be short. But the tool is also aimed to be used not only by usability expert but also by system designers who will perform the evaluation several times in the design process. But evaluation are only done at some steps in the design. So, the time between different uses of the editor can be long. Still longer if the editor is used on different design projects. To conclude, the period of remanence is to be long as a frequent use of this tool in all the design process is not likely.
- **Subjective satisfaction to use the system** : evaluators are used to perform evaluation on paper without software tool. Such a tool is not absolutely needed to perform an evaluation. So, if the subjective satisfaction perceived by users is low, if they do not perceive benefit from using the tool, or if using it is too complex in relation with what they are used to do, they probably will not use this editor in the long term. That's why this criterion is important to take into account.
- **Degree of coverage** : a high degree of coverage in the mechanisms of the interface is always preferable. But as the application we want to build is not crucial, the degree of coverage must be appropriate without being perfect.

### 5.3. Derivation of interaction styles

As we said that we wanted to reproduce in the interface of the editor, the different documents that are part of the objective environment of the prescribed



task, intuitively the style of interaction that we should choose in the « filling of form ».

## 6. Derivation of the functional specifications : Chaining graph of the functions

The chaining graph of the function is a multigraph where nodes are functions or messages and arcs are relation of precedence [BODART 98].

### 6.1. Task 1 : Enter the inputs of the CW

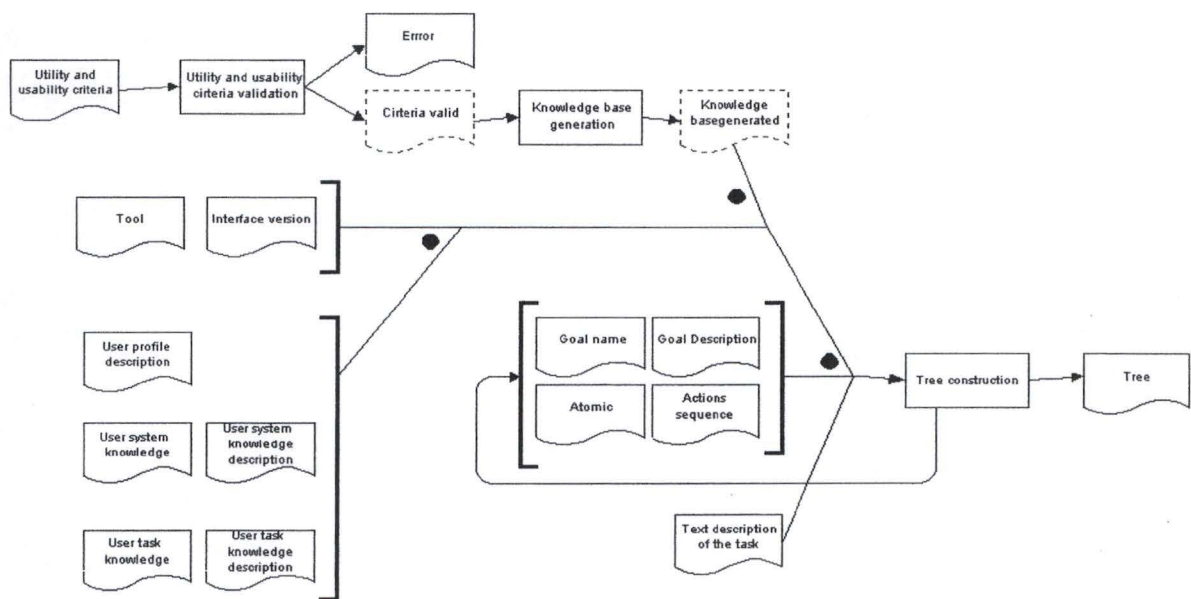


Figure 32 - Chaining graph of the functions : Enter the inputs of the CW

Legend			
	= internal information		= external information
	= AND group of information		= function
	= AND		= exclusive OR (XOR)

## 6.2. Task 2 : Performing the CW

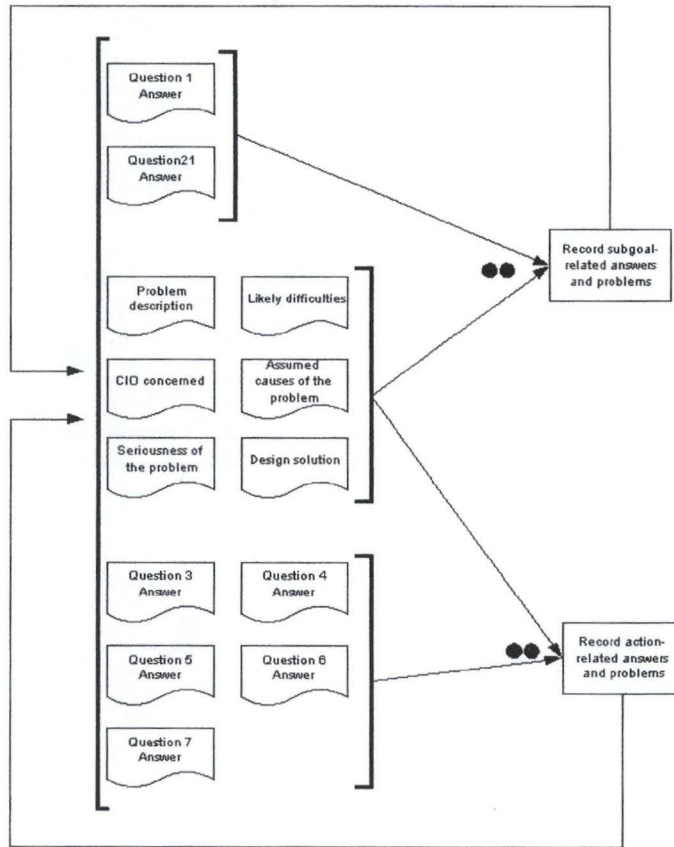


Figure 33 - Chaining graph of the functions : Performing the CW

### 6.3. Task 3 : Consult the problems list

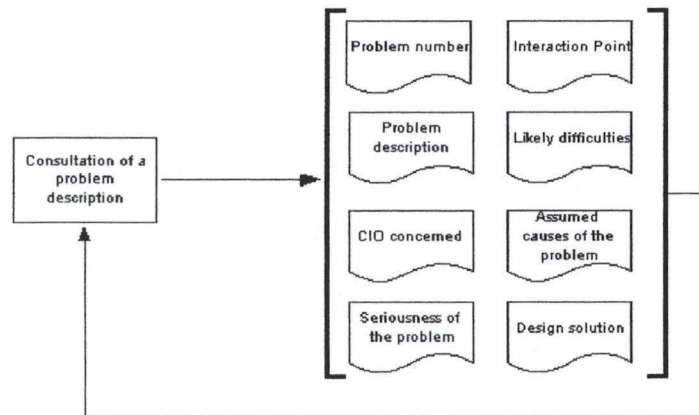


Figure 34 - Chaining graph of the functions : Consult the problem list

## 6.4. Task 4 : Use UAN Library

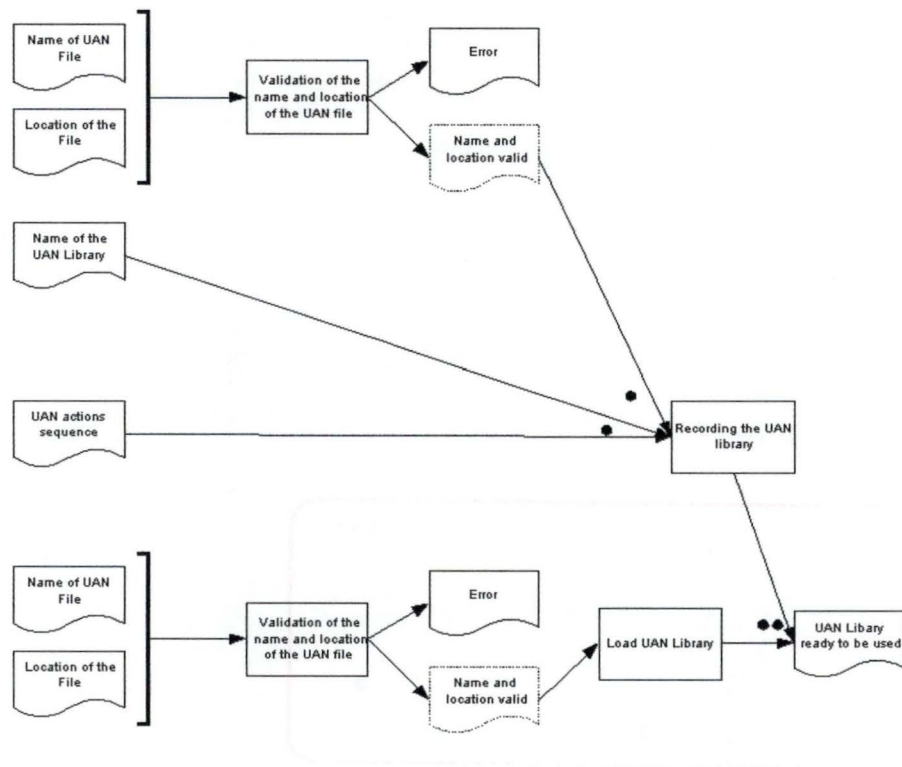


Figure 35 - Chaining graph of the functions : Use UAN Library

## 7. Definition of the presentation

### 7.1. Identification of the presentation units

From the different chaining graphs presented in the section 6, we can derive the presentation units, each of these corresponding to one of the subtask of the task considered.

#### 7.1.1. Task 1 : Enter the inputs of the CW

For the first task « Enter the inputs of the CW », three subtasks have been distinguished. One unit of presentation will be associated to each one. The three PU are named :

1. PU1 : Capture of the interface information
2. PU2 : Capture of the user profile
3. PU3 : Construction of the goal/subgoals tree

The dialogue attributes described in the section 5.1 are the same for each of the three PU.

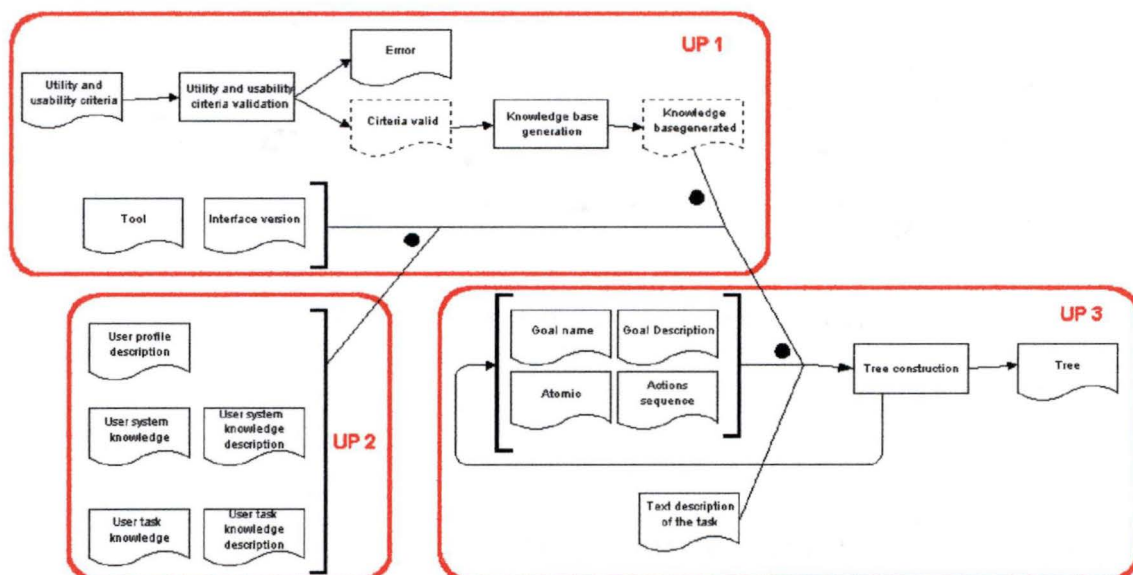


Figure 36 - Presentation Units : Enter the inputs of the CW

### 7.1.2. Task 2 : Performing the CW

Two subtasks of the task « Performing the CW » are similar : the first one is to answer the subgoal-related questions and the second one is the same subtask but for the action-related questions. That's why we identify only one presentation unit including all the functions, input and output messages of this second task. This PU name is « Capture of the information related to the performing of CW ».

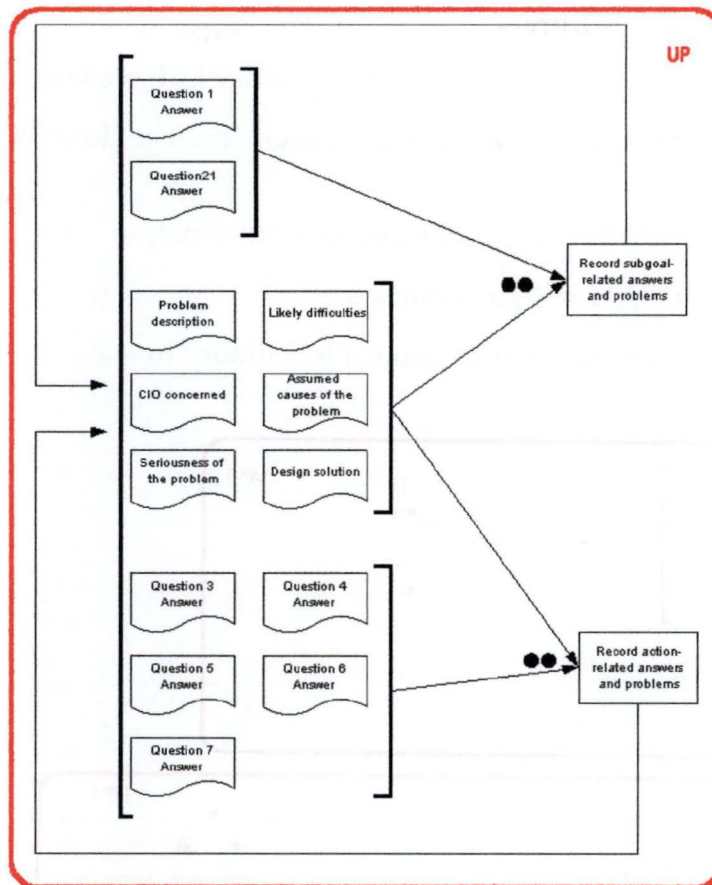


Figure 37 - Presentation Units : Performing the CW

### 7.1.3. Task 3 : Consult the problems list

As this third task is elementary, only one presentation unit is retained.

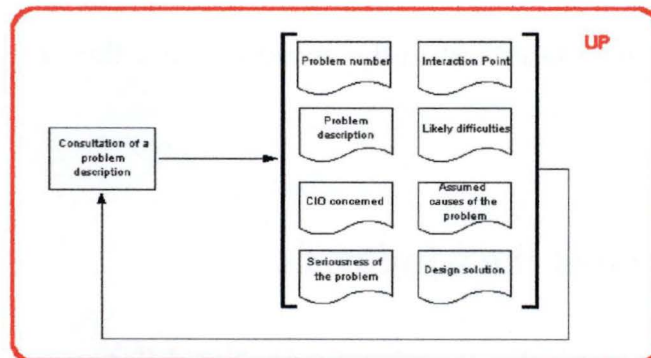


Figure 38 - Presentation Units : Consult the problems list

#### 7.1.4. Task 4 : Use UAN library

For the fourth task « Use UAN library », three subtasks have been distinguished. One unit of presentation will be associated to each one. The three PU are named :

1. PU1 : Capture of UAN libray name and save information .
2. PU2 : Capture of UAN actions sequence.
3. PU3 : Capture of the information related to a library to use.

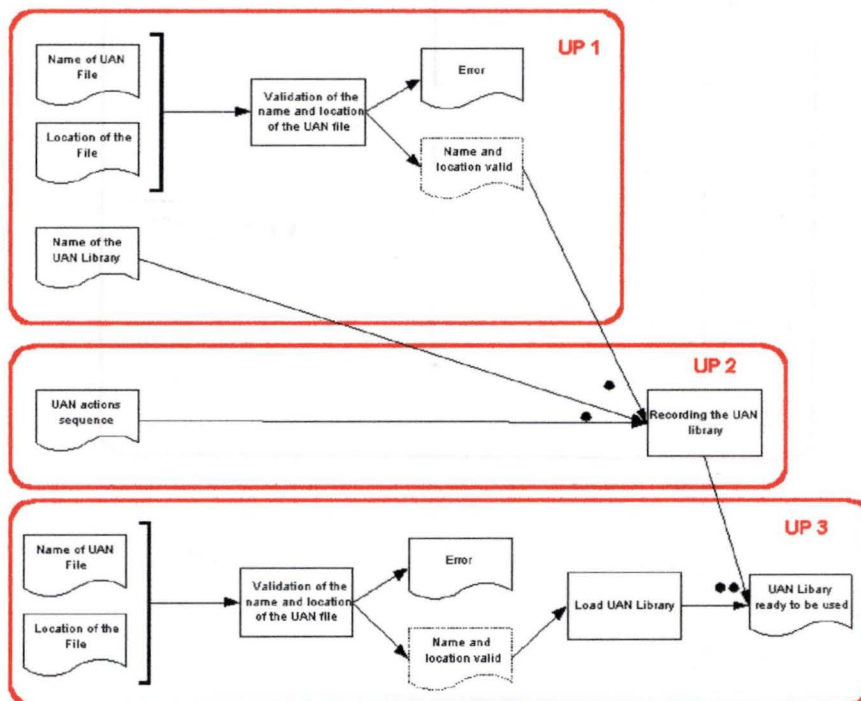


Figure 39 - Presentation Units : Use UAN Library

The dialogue attributes described in the section 5.1 are the same for each of the three PU.

## 7.2. Identification of the windows

In each PU identified on the chaining graph, the different windows are to be defined. The criterion retained for the identification of the windows is the « input/output » identification and the identification of group. The first one



means that the inputs are presented in a windows and the outputs in another one. The identification of group is used to group information in one window, for example for ergonomical reason. The combination of these two criteria give for the presentation units the decomposition in windows as shows in the following figure.

### 7.2.1. Task 1 : Enter the inputs of the CW

In each presentation unit, a window should be created for any unit of information. However, we will group input messages that are related to each other. For example, the information related to one subgoal should be capture in one time (window 32 in the following figure). In the same way, we create a unique window for what concerns the utility and usability criteria information.

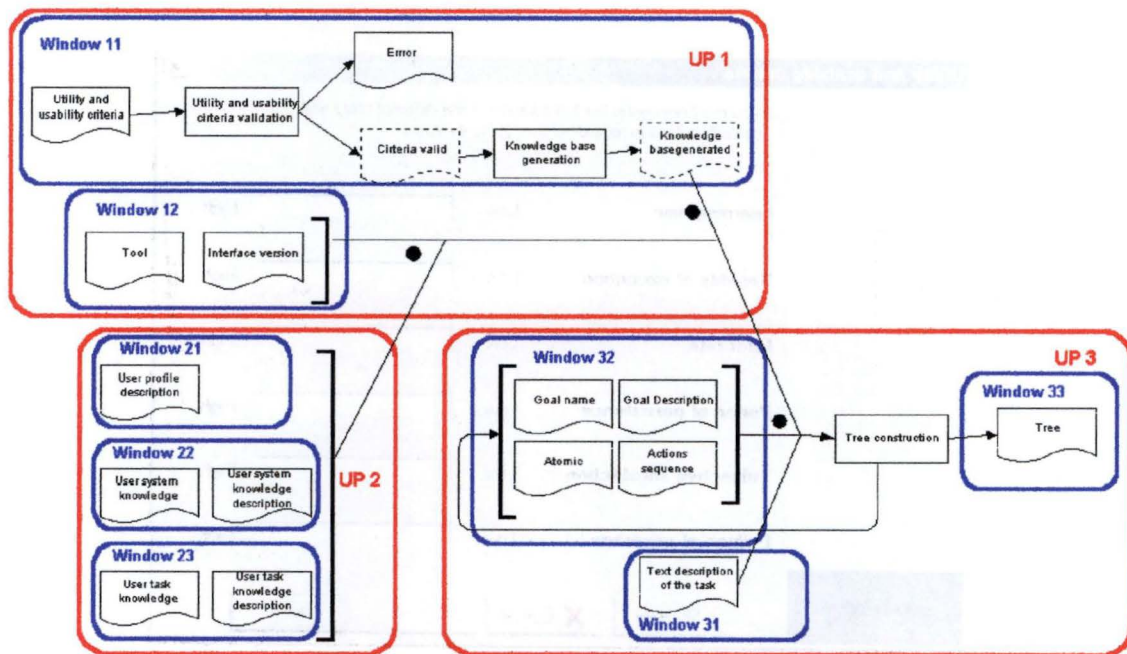


Figure 40 - Definition of the windows : Enter the inputs of the CW

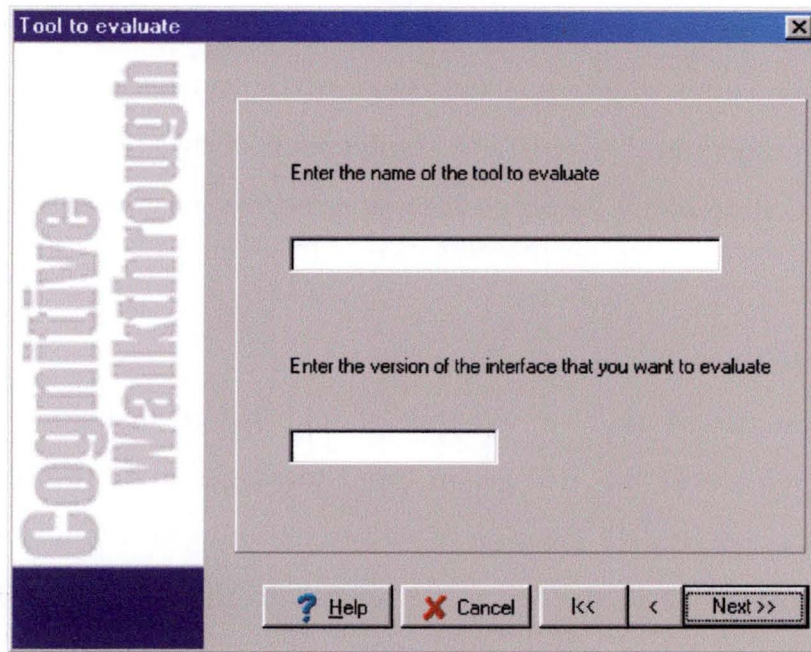


Figure 41 - Task 1 : Window11

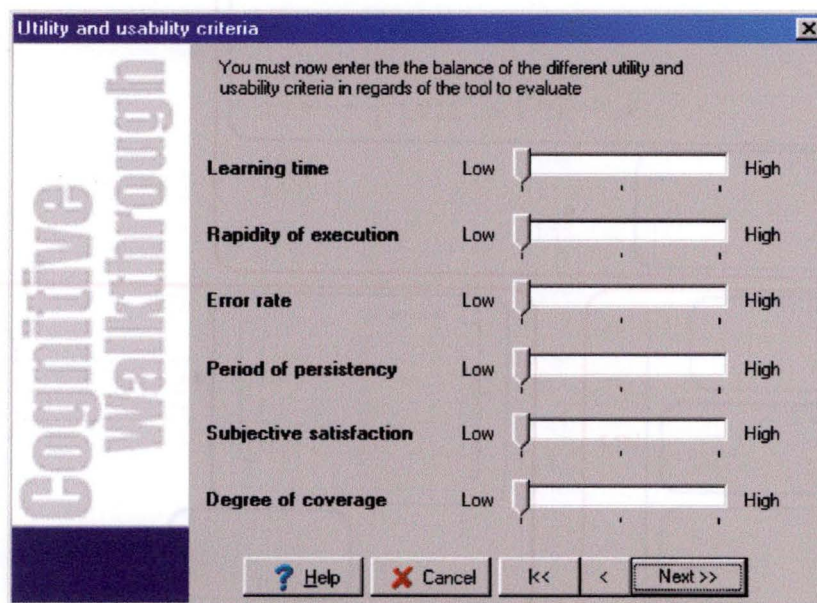


Figure 42 - Task 1 : Window12

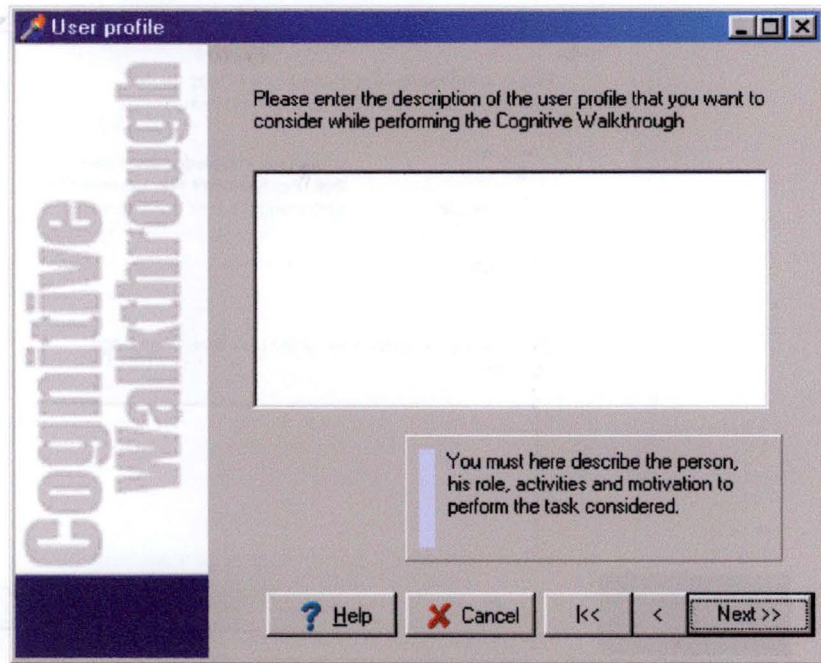


Figure 43 – Task 1 : Window21

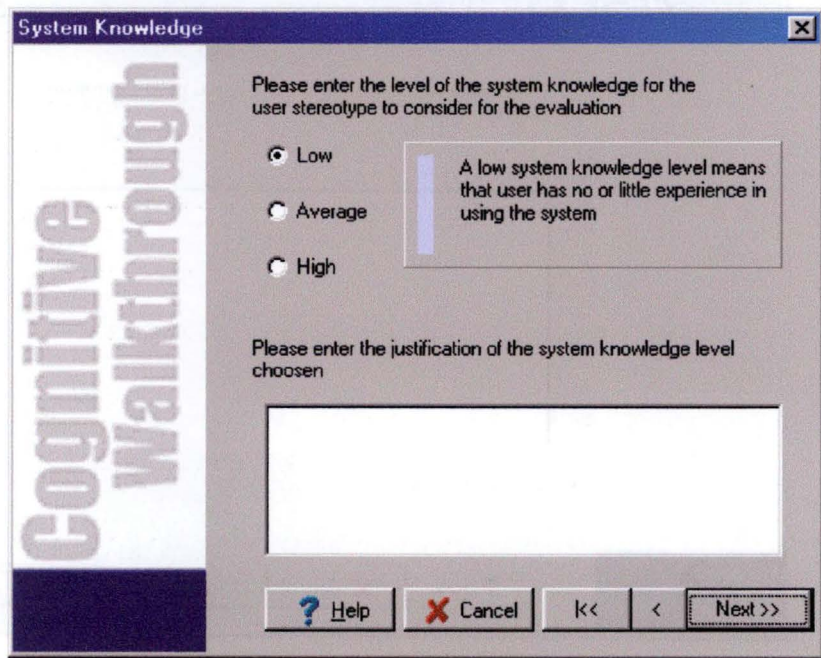


Figure 44 – Task 1 : Window22

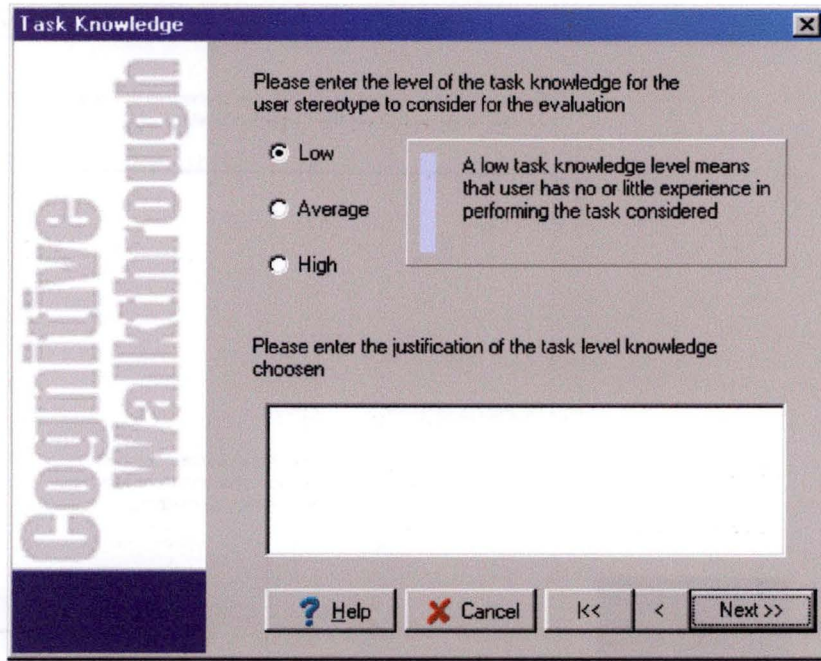


Figure 45 - Task 1 : Window23

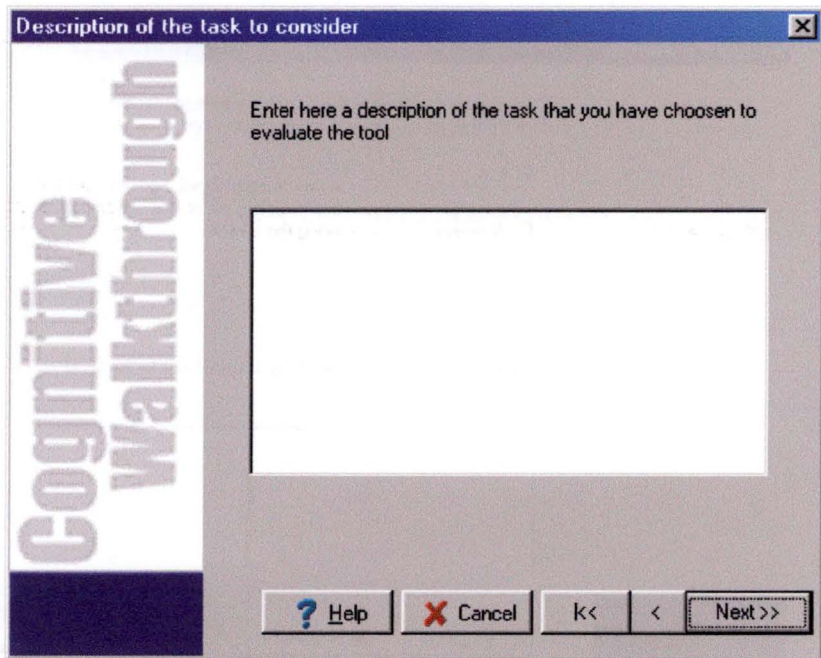


Figure 46 - Task 1 : Window31

**Capture of information about a goal** [X]

Fill in the name and the description to the new goal to insert :

Goal Name

Goal Description

Number of subgoals required to achieve this subgoal

Select the following option if this goal cannot be decomposed :  Atomic

Actions

	Action	Feedback	System Status

Figure 47 - Task 1 :Window32

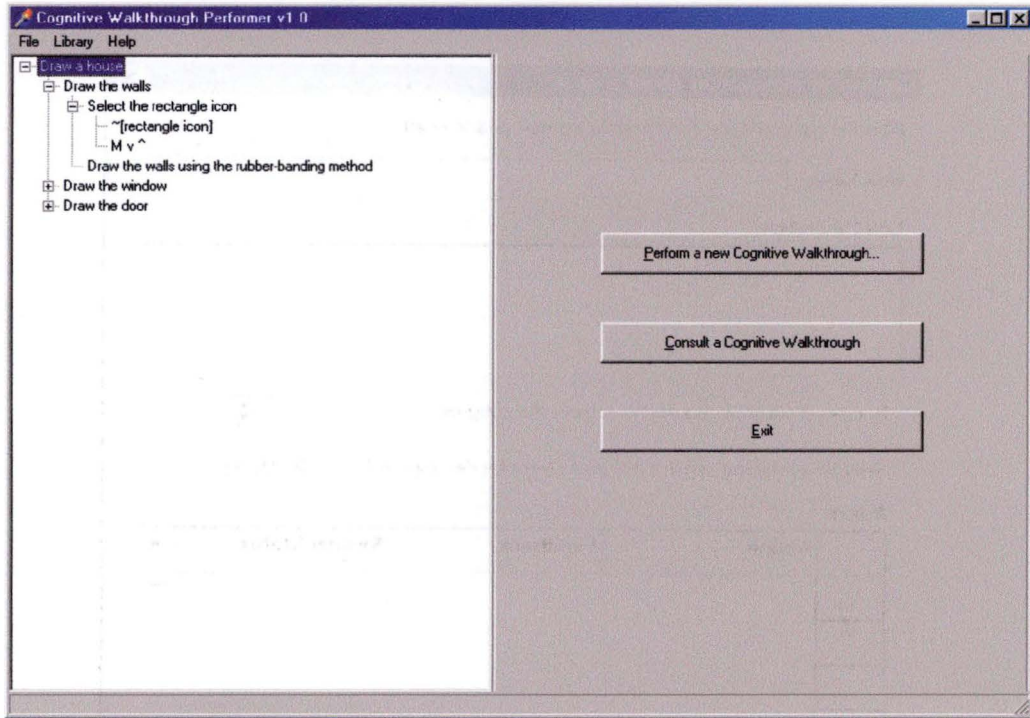


Figure 48 - Task 1 : Window33

### Extra window for UP3

This window is introduced to assure the guidance function. As we explained before, UAN libraries must be available to accelerate the capture of the actions (by drag&drop the UAN actions).

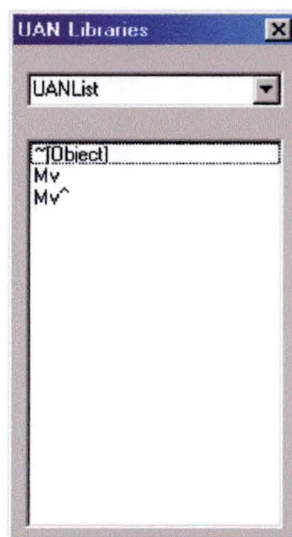


Figure 49 - Task 1 :Extra Window for a guidance service

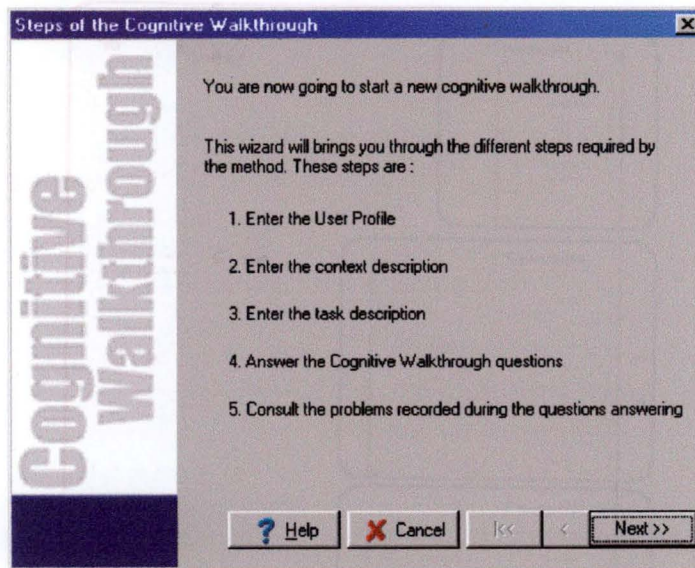


Figure 50 - Extra window inserted for a guidance service

The figure 50 has been built to take into account the results of the reduced CW performed on the projected task<sup>6</sup>. It aims to provide information about the different steps to accomplish a Cognitive Walkthrough.

### 7.2.2. Task 2 : Performing the CW

To respect the metaphor of form we said we will use, a separate window is created for the capture of the subgoal-related answers, of the action-related answers and of the problem description. This is likely to be a good definition of the windows as it is similar to the real task of performing the CW where the evaluator has separate forms to record these information.

<sup>6</sup> See section 3.5 of this chapter.

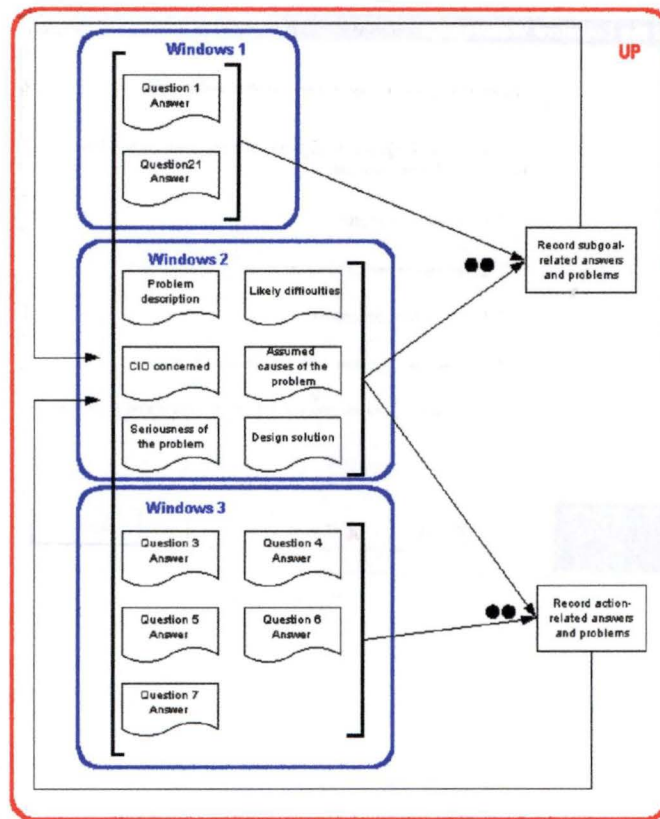


Figure 51 - Definition of windows : Performing the CW

Action-related questions

Action: \_\_\_\_\_

Question 3 : Can users notice that the correct action is available ?

Yes    Affirmative usual answers

No    Negative usual answers

Other

Question 4 : Will users associate the correct action with the subgoal they are trying to achieve?

Yes    Affirmative usual answers

No    Negative usual answers

If you found a problem about this action, record it    Record problem...

? Help    X Cancel    K<    <    Next >>    Finish

Figure 52 - Task 2 : Window 1



Record problem found

Identification of the problem

Problem Number  Interaction point

Description of the problem

What is the problem ?

What are the concrete difficulties that may be encountered ?

What are the assumed causes of the problem ?

Seriousness

Interface features involved in the problem

? Help X Cancel K< < Record next problem >> Finish

Figure 53 - Task 2 : Window2

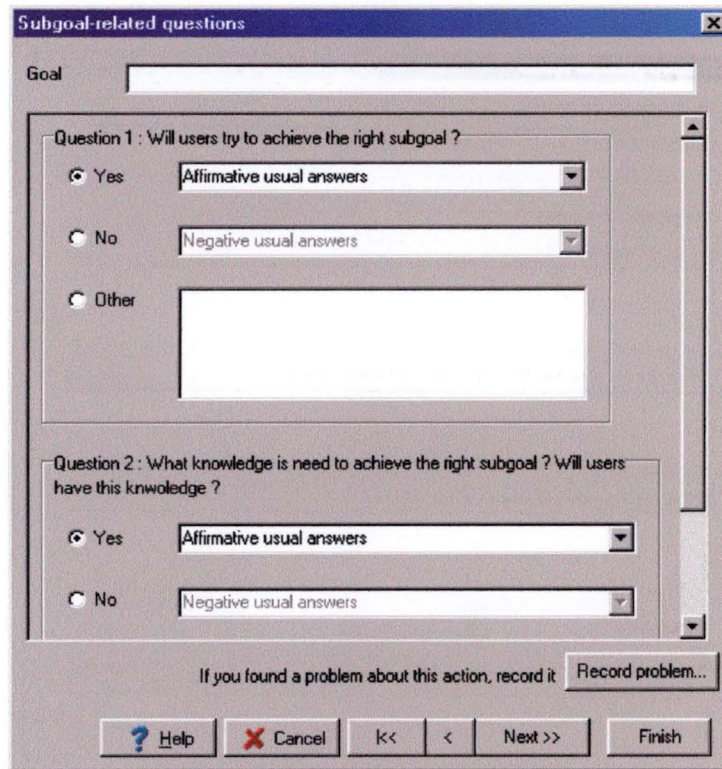


Figure 54 - Task 2: Performing the CW

### 7.2.3. Task 3 : Consult the problems list

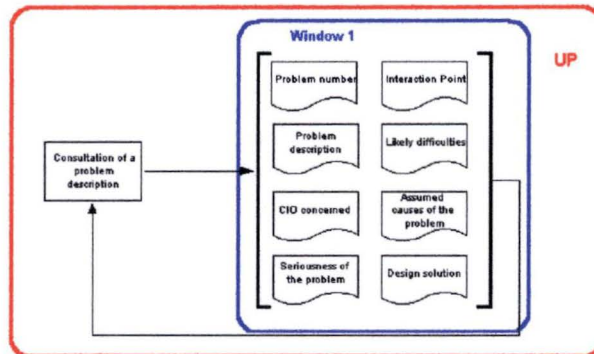


Figure 55 - Definition of windows : Consult the problems list

Description of a problem found

Identification of the problem

Problem Number  Interaction point

Description of the problem

What is the problem ?

What are the concrete difficulties that may be encountered ?

What are the assumed causes of the problem ?

Seriousness

Interface features involved in the problem

? Help X Cancel << < Next problem >>

Figure 56 - Task 3 : Window1

## 7.2.4. Task 4 : Use UAN Library

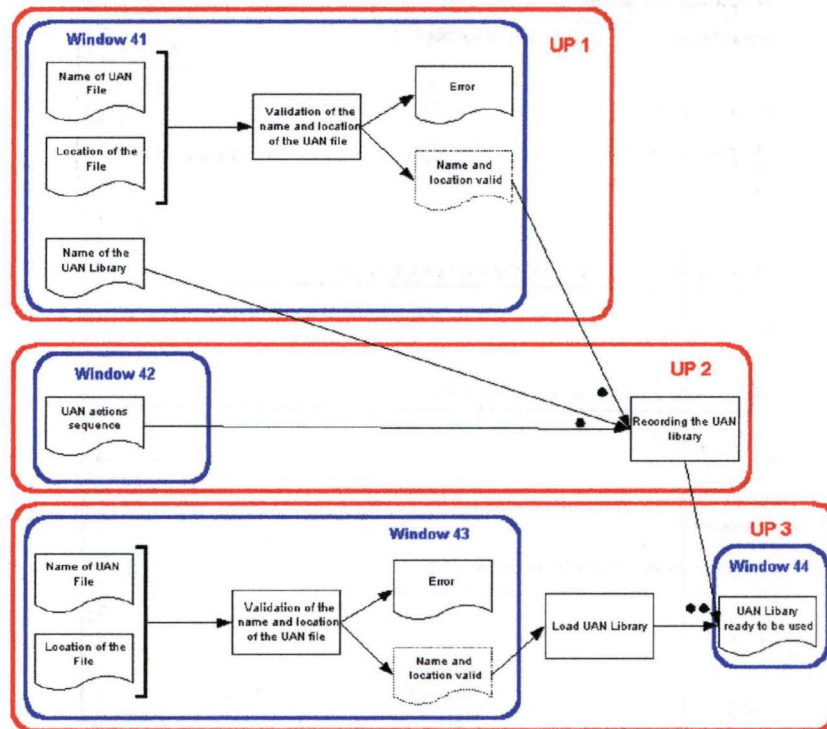


Figure 57 - Definition of the windows : Use UAN Library

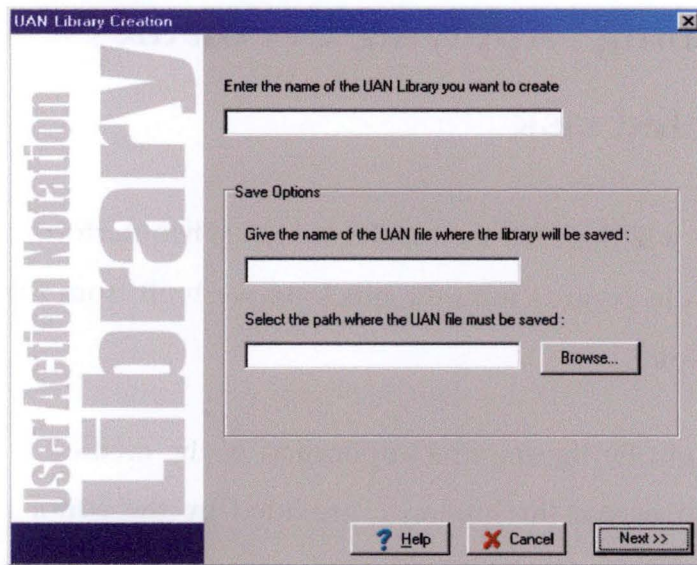


Figure 58 - Task 4 : Window41

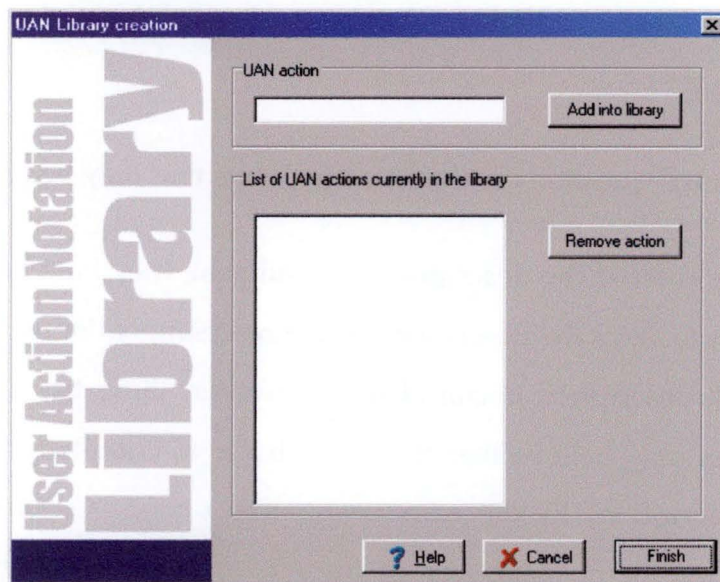


Figure 59 - Task 4 : Window42

## 8. Bootstrapping : Applying CW on the implemented task

In this section, we will describe the results we got when performing CW on the implemented tasks in regards with the mock-up we built from the task analysis realized in this chapter.

The remarks brought by the the first application of the method were taken into account. The result is that the guidance provided by the editor is strong. It is likely that there will not be any problems in regards with the goal structure of the task. Either the user has the knowledge of how to perform tasks, or the system provides the required information to let him know about the sequence of subgoals to achieve.

But the action-related questions revealed some points that may be problems.

When users have entered the description of a subgoal, they have to click on the « Next > » button to enter the description of the next subgoal or on the « Finish » button if the task description is completed. However, these two labels may be ambiguous. Users may believe that the « Finish » is to complete the capture of one subgoal description. In the same way, the « Next > » button may be interpreted as « go to the next step of the CW », i.e. answering the method questions. These button should be better labelled. A design solution would be to name the « Next > » button « Next subgoal > » and the « Finish » button « Finish the decomposition ».

Still concerning this window, the UAN Library are available by clicking on the button « UAN Libraries » but if users don't know about UAN, this will not make sense for them. A text help should be included to shortly explain what this means. Further when the UAN Libraries window appears, the UAN actions can

be inserted into the sequence of actions of the subgoal by drag&drop. But no indication about this way of proceeding is given. This should be added in the UAN Libraries Window.

In relation with the task « Performing the Cognitive Walkthrough », when users have to choose answer to give, they have three choices : Yes, No or Other. When choosing one of the two first, they have to select an usual answer associated with the question. These three choices are exclusive. But users may want to add comments to detail the answer they have given (YES or NO). So, a design solution for this problem should be to offer three exclusive choice : YES, NO and PERHAPS. Moreover, the opportunity to insert a comment for each question must be let to users. We could add a checkbox that will, once checked, enable the text capture.

Finally, the last problem that CW revealed was that while performing the method, the name of the subgoal of action is given as well as its order identifier (e.g. 1.1.2.). But users may be quickly lost in all the performing : they surely want information about where they are in the task decomposition. To indicate the progress obtained when switching from one subgoal (action) to another subgoal (action), the graphical tree of the decomposition must be always seen with the current action or subgoal highlighted.

So, Cognitive Walkthrough revealed some problems that must be now taken into account. After the implementation of the design solutions proposed here, another iteration of CW should be applied to evaluate the next version of the editor interface.

## 9. Conclusion

This chapter was interesting in two ways. Firstly, it specifies a complete task analysis walkthrough to build an editor supporting the Cognitive Walkthrough task. This analysis should be a good guide for a future implementation of this tool. In Appendix C, we provide a description of the data conceptual model of the information system, that should be associated with this CW editor, and the treatments conceptual model. We also added a global architecture and a specification of the treatments module.

The second contribution of this chapter is to give a practical example of the inclusion of the Cognitive Walkthrough method into a global design process. Even it is a small and incomplete one !



## *C o n c l u s i o n*

We have tried, along this thesis, to show the importance of the insertion of the assessment of an interface into the a lifecycle of its design process.

Indeed, for a lot of people, it seems to be easy to create good interfaces : some good sense and the work is done. Perhaps we thought this before getting in touch with the usability problematic. However... How many interfaces can be qualified to interfaces of quality ?

The taking into account of the quality of an interface in the design process is vital and even if the good sense can help, a rigorous walkthrough is needed to try to reach a so much desired quality of interface.

In this work, we have presented one method of interface evaluation : the Cognitive Walkthrough. The placement we made in the Informatic Center of the University of Sunderland brought us valuable and practical experience in evaluating an interface, not only with Cognitive Walkthrough but also with other usability inspection methods.

The two main contributions we tried to bring were to give a theoretical foundation of this method and to locate it in a design process we have learned in the second master degree : the task analysis walkthrough. About the location, we have tried to place the method not only for an already-made interface but also very soon in the design process to permit a rapid review of the interface, before the availability of any mock-up or prototype .

The lecture we have studied in second's master degree already takes in consideration the quality of an interface and its evaluation but by other means : ergonomic rules and design criteria. The CW method is a task centered method. We think this thesis could complete the lecture of Professor Bodart.

We have learned some principles after this introspection in the heart of the assessment of interfaces. No method is perfect and can find out all the usability problems of an interface or a future one.

A possible solution would be to combine different methods as task-centered method and heuristic one. A future work is maybe to evaluate the method « Heuristic Walkthrough » which perform this combination.

However, it's important not to leave the economic aspect out. The CW method is already a time-consuming method. Added to another one, what will happen ?

We think it will be good to take into consideration the combination of different methods but also the nature of interface which will maybe determine the kind of method to apply on it, ie. Web interfaces may require different evaluation from the evaluation of a interface based on Virtual Reality.

## BIBLIOGRAPHY

### [BODART 98]

François Bodart (1998). Cours Introductif à la Conception des Interfaces Homme-Machine, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur.

### [FARENC 97]

Christelle Farenc (1997). Ergoval : une méthode de structure des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques, Ph.D thesis, Université de Toulouse 1.

### [FRANZKE 95]

Marita Franzke (1995). Turning research into practice : characteristics of display-based interaction. *Human Factors in computing systems - CH'95 Conference Proceedings*. 421-428.

### [HOM 98]

James Hom (1998). The Usability Methods Toolbox.

Available at <http://www.best.com/jthom/usability/>

### [HIX, HARTSON 93]

Deborah Hix and H. Rex Hartson (1993). Developing user interface - Ensuring usability through products and process, Wiley. 152-220.

### [ISO 92]

ISO/WD 9241 - Part 11 Ergonomic requirements for Office Work with Visual Displays Units, International Standard Organization (1992).

[JEFFRIES et al. 91]

Robin Jeffries, James R. Miller, Cathleen Wharton and Kathy M. Uyeda (1991). User Interface Evaluation in the real world : A comparison of four techniques. *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*. 119-124.

[JOHN, KIERAS 94]

Bonnie E. John, David E. Kieras (1994). The GOMS Family of Analysis Techniques : Tools for Design and Evaluation, Carnegie Mellon, University School of Computer Science. Technical Report CMU-CS-94-181.

Available at <ftp://reports.adm.cs.cmu.edu/usr/anon/1994/CMU-CS-94-181.ps>

[JOHN, PACKER 95]

Bonnie E. John, Hilary Packer (1995). Learning and using the Cognitive Walkthrough method : A case study approach. *Human Factors in computing systems - CH'95 Conference Proceedings*. 429-436.

[JOHN, MASHYNA 97]

Bonnie E. John, Matthew Mashyna (1997). Evaluating a Multimedia Authoring Tool. *Journal of the American Society for Information Science* 1004-1022.

[JOHNSON 92]

P. Johnson (1992). *Human Computer Interaction*, Mac Graw-Hill.

[LAVERY, COCKTON 96]

Darryn Lavery and Gilbert Cockton 1996. Heuristic Evaluation Usability Evaluation Materials. Technical Report TR-1996-15, University of Glasgow. Available at <http://www.dcs.gla.ac.uk/asp/materials/>.

[LAVERY, COCKTON 97]

Darryn Lavery and Gilbert Cockton 1997. Cognitive Walkthrough Usability Evaluation Materials. Technical Report TR-1997-20, University of Glasgow. Available at <http://www.dcs.gla.ac.uk/asp/materials/>.

[LEWIS 93]

Clayton Lewis and John Rieman (1993). Task-centered user interface design, A practical introduction.

Available at <ftp://ftp.cs.colorado.edu/pub/cs/distrib/clewis/HCI-Design-Book>

[LEWIS 97]

Clayton Lewis, Cathleen Wharton (1997). Chapter 30 : Cognitive Walkthroughs. In *Handbook of Human-Computer interaction*, Elsevier Science B.V.. 717-730.

[NIELSEN 93]

Jakob Nielsen (1993). Chapter 5: Usability Heuristics. In *Usability Engineering*, Academic Press. 113-163.

[NIELSEN 94]

Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods*. John Wiley & Sons, New York, NY.

[NORMAN, DRAPER 86]

Noman D.A., Draper S.W (1986)., *User Centered System Design on Human Factors Interaction : New Perspectives*. Lawrence Erlbaum Associates Publishers, Hillsdade.

[SEARS 97]

Andrew Sears (1997). Heuristic Walkthroughs : Finding the Problems Without the Noise. In *International Journal of Human-Computer Interaction*, Lawrence Erlbaum Associates, Inc. 213-234.

[SCHNEIDERMAN 92]

Schneiderman B. (1992), *Designing the User Interface : Strategies for Effective Human-Computer Interaction*, 3<sup>rd</sup> Edition. Addison-Wesley,.

[VANDERDONCKT 93]

Jean Vanderdonckt (1993). Corpus ergonomique minimal des applications de gestion, Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur.

[WOOD 98]

James Wood (1998). GOMS Introduction. Available at <http://www.usabilityfirst.com/GOMS/>.

[WOODWARD 98]

Bill Woodward (1998). Evaluation Methods in usability Testing, CS5326. Available at <http://www.swt.edu/~hd01/5326/projects/BWOODWARD.HTML>.

[WHARTON et al. 92]

Cathleen Wharton, Janice Bradford, Robin Jeffries and Marita Franzke (1992).  
Applying Cognitive Walkthroughs to more complex user interfaces :  
experiences, issues, and recommendations. *Human Factors in computing  
systems - CH'92 Conference Proceedings*. 381-388.

[WHARTON et al. 94]

Cathleen Wharton, John Rieman, Clayton Lewis and Peter Polson (1994).  
The Cognitive Walkthrough : A Practitioner's Guide, In Nielsen, J. and  
Mack, R.L. (Eds.), *Usability Inspection Methods*. John Wiley and Sons, New  
York. 105-140

# APPENDIXES



## TABLE OF CONTENTS

### **APPENDIX A - THE USER ACTION NOTATION LANGUAGE ..... 3**

1.	INTRODUCTION.....	3
2.	PURPOSE OF UAN.....	4
3.	DESCRIPTION OF UAN .....	4
4.	MORE ON UAN ACTIONS .....	6
5.	EXTENSIBILITY OF UAN .....	8
5.1.	Mouse pointers.....	8
5.2.	Hierarchical objects.....	10
5.3.	Functions .....	12
5.4.	The Keys .....	14

### **APPENDIX B - THE CONGNITIVE WALKTHROUGH : AN EXMAPLE OF USE ..... 15**

1.	INTRODUCTION.....	15
2.	INPUTS OF THE COGNITIVE WALKTHROUGH.....	15
2.1.	A task to analyse .....	15
2.2.	The abstract task.....	17
2.3.	User profile.....	18
2.4.	Context of work .....	18
2.5.	Utility and usability criteria.....	18
2.6.	The concrete task (the implemented task).....	19
3.	PERFORMING THE COGNITIVE WALKTHROUGH.....	29
4.	RESULTS OF THE METHOD AND THEIR INTERPRETATION .....	56
5.	CONCLUSION.....	74

### **APPENDIX C - SPECIFICATION OF THE COGNITIVE WALKTHROUGH EDITOR..... 75**

1.	INTRODUCTION.....	75
2.	DATA CONCEPTUAL MODEL OF THE INFORMATION SYSTEM .....	75
2.1.	ERA model of the Information System.....	75
2.2.	Data dictionary .....	77
3.	TREATMENTS CONCEPTUAL MODEL OF THE IS .....	81
4.	GLOBAL ARCHITECTURE.....	88
5.	SPECIFICATION OF THE TREATMENT MODULES.....	90

## TABLE OF FIGURES

Figure 1 - UAN representation of mouse pointers .....	9
Figure 2 - Object properties hierarchy .....	11
Figure 3 -- The hint text of an icon.....	12
Figure 4 - A help text in the Status Bar.....	13
Figure 5 - Pop-up Menu associated to a Menu Item.....	13
Figure 6 - CW example : Screenshot 1 .....	16
Figure 7 - CW example : Screenshot 2.....	17
Figure 8 - CW example : Screenshot 3.....	57
Figure 9 - CW example : Screenshot 4.....	58
Figure 10 - CW example : Screenshot 5 .....	60
Figure 11 - CW example : Screenshot 6 .....	62
Figure 12 - CW example : Screenshot 7 .....	63
Figure 13 - CW example : Screenshot 8 .....	69
Figure 14 - ERA model of the Information System.....	76
Figure 15 - Global architecture of the CW editor.....	89

## THE USER ACTION NOTATION LANGUAGE

### **1. Introduction**

The User Action Notation language was introduced by Hartson, Siochi and Hix in 1990. It was developed to respond to a concrete need. Using prose descriptions to describe the behaviour that an interface should have is not a trivial task : these descriptions are usually inaccurate, sometimes ambiguous and misleading. Moreover, the implementers who have to read these prose specifications and understand them to translate them into a concrete interface. If details are omitted or not clearly understood by the implementers, they will have to make some guess about what the interface designer wanted. UAN was created to eliminate these problems.

In our work, we use it with the same state of mind. Tired of using prose to describe the sequence of actions, feedback and system status in the task description, UAN brought us a quicker way to realize the task description. Of course, all the participants of the design must be able to use UAN : the interface designer who specifies the interaction, the implementers and the usability evaluator.

The present appendix has not the pretention to offer a full description of the UAN language but is only aimed to give keys of understanding of the UAN language in order to permit a comprehensive reading of the task description

presented in the Cognitive Walkthrough example (Appendix B). For instance, UAN offers also ways to describe tasks sequence, task interruption, and so on. As we used UAN only to describe actions, we will not present here these last features of this language.

## 2. Purpose of UAN

The goal of UAN is to permit the recording of the interface interaction in a precise, concisely, unambiguous and detailed way. Further, as being a standardized way of describing interaction, it can help people joining a development team to minimize the misunderstanding that could arise from prose descriptions (because they have not followed all the design process).

« UAN is a user- and task-oriented notation that describes physical actions and behavior of the user and interface as they perform a task together » [HIX 93].

## 3. Description of UAN

To get you quickly in touch with this language, we will introduce a first simple example. Here is the prose description of selecting a file icon.

1. Move the mouse cursor over the file icon
2. Depress and release the left mouse button.

The UAN description of these actions is as follows :

1. ~[File icon' ]
2. Mv^

The ~ character means moving the mouse cursor. The destination of the cursor is indicated into the brackets. The ` character means that we refer to a specific file icon and not to any file icon. The tilde (~) is used to give the impression of motion. The second action  $Mv^{\wedge}$  denotes the depressing (v) and the releasing (^) of the mouse button (M).

At this step, we only described the user's actions. But we said before that UAN was used to described not only user's actions but also the system feedback and state. This lack is fullfill below. The system considered is Microsoft Word97.

Clicking on the file icon		
User action	System feedback	System status
~[File icon']	File icon'!	
$Mv^{\wedge}$	<ul style="list-style-type: none"> <li>• File icon'-!</li> <li>• File icon'!!</li> </ul>	Selected = selected U {file icon'}

File icon'! indicates that feedback is given by the system when the mouse cursor is over the file icon. At the second action level, File icon'-! means that the first feedback is over and File icon'!! that a second feedback, different from the first one, is shown. When clicking on the icon, the system status changes because the file icon is now selected.

This was a simple example that can be complexify much more. Indeed, usually when clicking on an object, the other objects that were selected are not anymore. This information feedback should also be included in our description. This can be simply written by :

$\forall$  file icon  $\neq$  file icon' : file icon-!

Intuitively, we can translate this by « for all the file icon different from the one selected, the highlighting is over ».

Note that the UAN description is a bit more complex.

Condition : {action}<sup>+</sup>

The first part of this UAN expression (before the « : » character) is the condition. The second part contains the set of actions (1-N) to trigger off when the condition is respected.

Our example becomes now :

Clicking on the save icon		
User action	System feedback	System status
~[File icon']	File icon'!	
Mv^	<ul style="list-style-type: none"> <li>• File icon'-!</li> <li>• File icon'!!</li> <li>• <math>\forall</math> file icon <math>\neq</math> file icon' : file icon-!</li> </ul>	Selected = selected U {file icon'}

## 4. More on UAN actions

Now that you are familiar with the UAN philosophy, this section will give you more knowledge about the actions that composes the base of the UAN language. There is a non exhaustive list of UAN actions.

~[x,y]

Move the mouse cursor to some arbitrary point  $x,y$  on the screen.

**Action\***

Like in regular expression, the start means that the action is repeated zero or more times. For instance,  $\sim[x,y]^*$  means that the mouse cursor can be moved on the screen to any points zero or more times.

**Action+**

Similar to the previous one, except that the action must be repeated at least one time.

**Object > ~**

Moving an object on screen (for example, dragging a file icon): the object follows the mouse cursor.

**Object >> ~**

Rubber-banding action. For instance, after a user has pressed the mouse button on a window corner and wants to resize it, he is grabbing the window with the cursor: `window' >> ~`

**@(x,y) display (object)**

The object is displayed at any coordinates of the screen. If the object has to appear at specific coordinates, it must be indicated by using  $@(x',y')$ . You can also use specific value for  $x$  and  $y$ .

**Erase(object)**

This means that the object specified is not displayed anymore.

**K « abc »**

User action that represents the capture of the string `abc` thanks to the Keyboard. A regular expression can be specified in place of the fixed string.

**K(xyz)**

Entering a value for the variable `xyz` via the Keyboard.

## 5. Extensibility of UAN

The power of UAN comes also from its possibilities of extension. In regards with the needs of the interaction designer to specify the interface, symbols can be replaced by some which are more explicit for the designer, other symbols or functions can be added. The only restriction is to let the language as intuitive as possible in order to be easily understood by all the design participants.

This section presents the UAN notations used in the task descriptions we made. These are not part of this original UAN notations set. We have mainly used these extensions to describe actions in softwares such as Microsoft Word97 and Microsoft PowerPoint95. But some of these extensions can be used to depict actions of other kind of softwares.

### 5.1. Mouse pointers

The mouse pointer gives always information either in indicating the action available at a given moment or in showing progress in the current task. Thus, the number of different mouse pointer shapes is high. In this base version of UAN, there is no representation of this kind of feedback. We propose below a representation for each pointer shape mainly based on a graphical similarity or a shortening of the pointer shape name.

















Pointer	Name <sup>1</sup>	Notation <sup>2</sup>
	I-beam pointer	I

<sup>1</sup> These pointers were found in "The GUI guidelines", Microsoft Press

<sup>2</sup> All the UAN notations are shown in "Courier" font



---

	Italic I-beam pointer	/
	Left-arrow pointer	<-
	Right-arrow pointer	->
	Cross-hair pointer	+
	Horizontal split pointer	^=v
	Vertical split pointer	<   >
	Horizontal outline pointer	<->
	Vertical outline pointer	^ v
	Four-headed outline pointer	4->
	Magnifying pointer	Zoom+
		Zoom-
	Left diagonal pointer for frames	^\v
	Right diagonal pointer for frames	v/^
	Graphics repositioning pointer	<-&4->
	Drag-and-drop pointer for moving	D&d
	Drag-and-drop pointer for copying	D&d+
	Downward-pointing arrow	-v

---

Figure 1 - UAN representation of mouse pointers

Sometimes the mouse pointer changes automatically when it is in some areas of the interface and returns back to its original shape when leaving those. Here follows some descriptions to explain these changes :

- When the mouse pointer moves over a toolbar : `pointer.shape1= <-`
- When the mouse pointer leaves a toolbar : `pointer.shape=pointer.shape0`
- When the mouse pointer moves over the left margin of the document :  
`pointer.shape1= ->`
- When the mouse pointer leaves the left margin of the document :  
`pointer.shape=pointer.shape0`

## 5.2. Hierarchical objects

It can be interesting to use some object properties in the task description. For instance, how can you say that the user moves the mouse pointer to the upper left handle of a shape? How can you indicate that an object ghost expands? A solution is provided by using a hierarchy of properties. Each object has got some properties which can be further decomposed into other properties.

### Example

Objects in drawing editors may have the properties showed in the following figure.

The ghost of an object is its outline. For instance, when moving an object in a drawing editor, the object is « dashed ». It is a feedback to inform that the user is currently moving the object. The handle of an object is what is used to manipulate it.

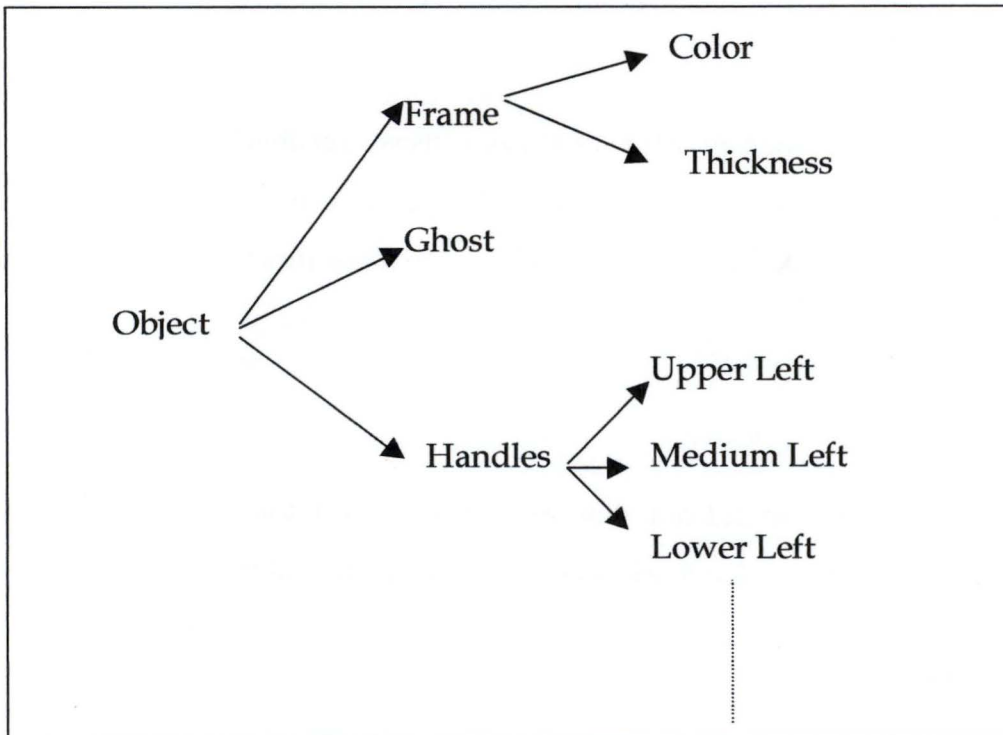


Figure 2 - Object properties hierarchy

### Example

To point the frame :

`Object.frame`

To point the ghost :

`Object.ghost`

To point the handles :

`Object.ulh3`  
`Object.mlh`  
`Object.llh`  
`Object.umh`  
`Object.urh`  
`Object.mrh`

---

<sup>3</sup> Upper Left Handle

Object.lrh  
Object.lmh

### 5.3. Functions

Functions are used when an action of a user causes a feedback linked to a specific object. For instance, when you click on a Menu Item, the appropriate menu or dialog box is displayed on the screen. That's what the functions are supposed to render.

#### Wait(time)

The function Wait is to indicate that the user is idle for some time. We used this mainly to express that a feedback occurs after some time of user inaction.

#### Hint(string)

The function Hint returns the hint text of a specific icon. The following figure illustrates this.



Figure 3 – The hint text of an icon

In UAN, this feedback is transcribed by `hint("Text Box")`.

#### ShowStatusBar(string)

The function ShowStatusBar return the help text for an icon in the StatusBar (fig. 4).

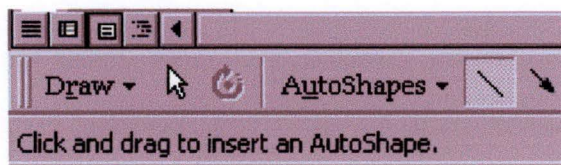


Figure 4 - A help text in the Status Bar

In UAN, this feedback is transcribed by `ShowStatusBar("Click and drag to insert an AutoShape")`.

### ShowMenu(MenuItem)

The function ShowMenu displays the pop-up menu associated with the Menu Item selected (fig.5).

### In

The In function is used only to know if an object is included in another one (e.g.: if shape is included in a selection rectangle ghost). Here is the UAN description of this function :

```
Object1.ul > Object2.ul & Object1.lr < Object2.lr : Object1
in Object2
```

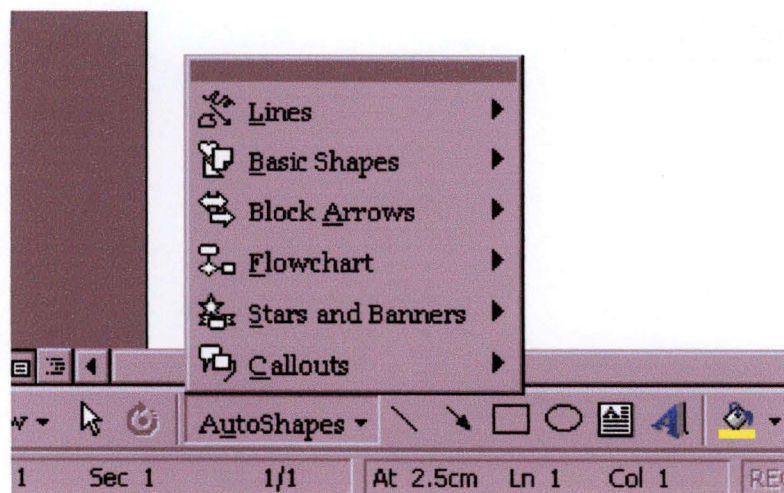


Figure 5 - Pop-up Menu associated to a Menu Item

## 5.4. The Keys

We only propose a notation for three labelled keys - those we needed - but it can be extended, in the same way, to all the others. Pressing and depressing the keys are made exactly like pressing and depressing the mouse.

Key	Pressing	Releasing	Clicking
SHIFT	SHIFT <sub>v</sub>	SHIFT <sup>^</sup>	SHIFT <sub>v</sub> <sup>^</sup>
CTRL	CTRL <sub>v</sub>	CTRL <sup>^</sup>	CTRL <sub>v</sub> <sup>^</sup>
DEL	DEL <sub>v</sub>	DEL <sup>^</sup>	CTRL

## THE COGNITIVE WALKTHROUGH : AN EXAMPLE OF USE

### **1. Introduction**

We will now present an example of the Cognitive walkthrough method following the way we have presented it in this thesis. We have used partly the User Action Notations to describe the actions.

### **2. Inputs of the Cognitive Walkthrough**

This is the final interface of the drawing editor in Microsoft PowerPoint 97 English Version.

#### **2.1. A task to analyse**

Here is a description of the task : The user starts from an opened slide (screenshot 1). The drawing toolbar is active and the user has to achieve some modifications on this slide :

- He has to rotate the two rectangles through an angle of 60 degrees
- He has to draw a no-filled square below the rotated rectangles

- He has to put the word : « Hello ! » in the south-east of the square

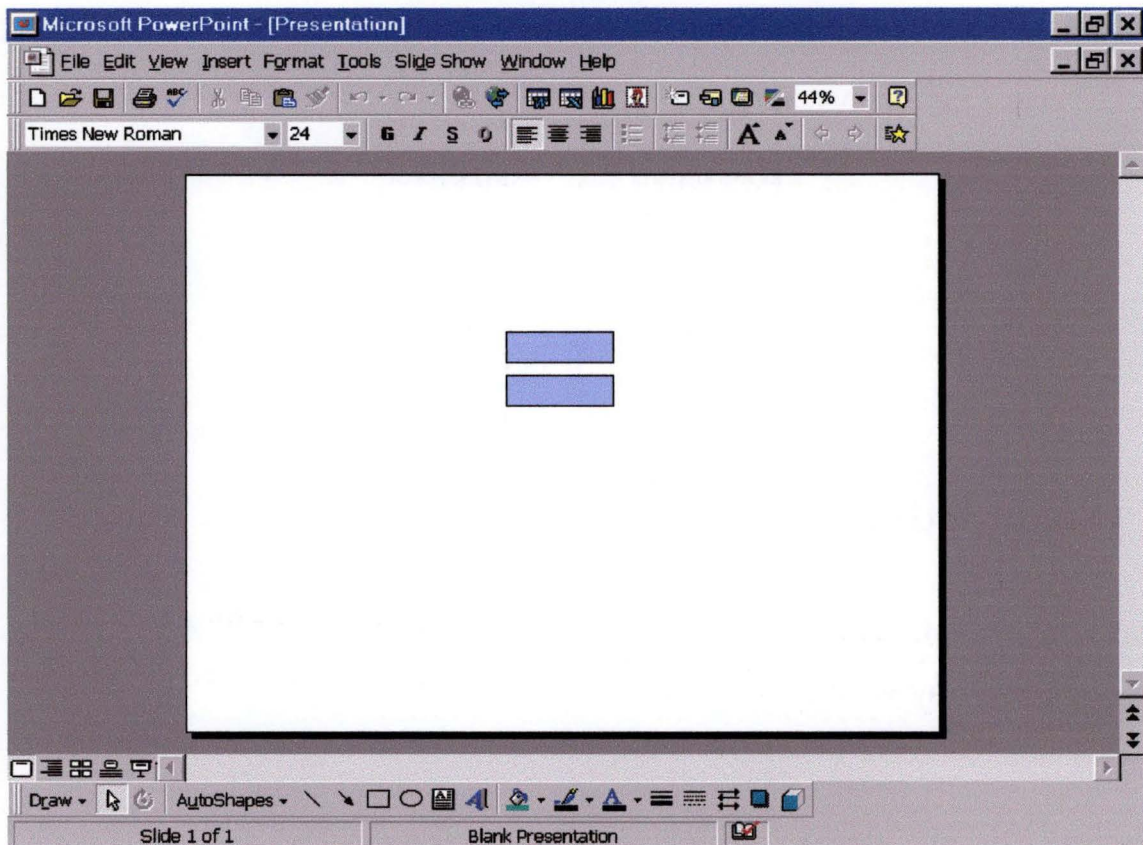


Figure 6 – CW example : Screenshot 1

Finally the slide has to resemble to this in the screenshot 2.



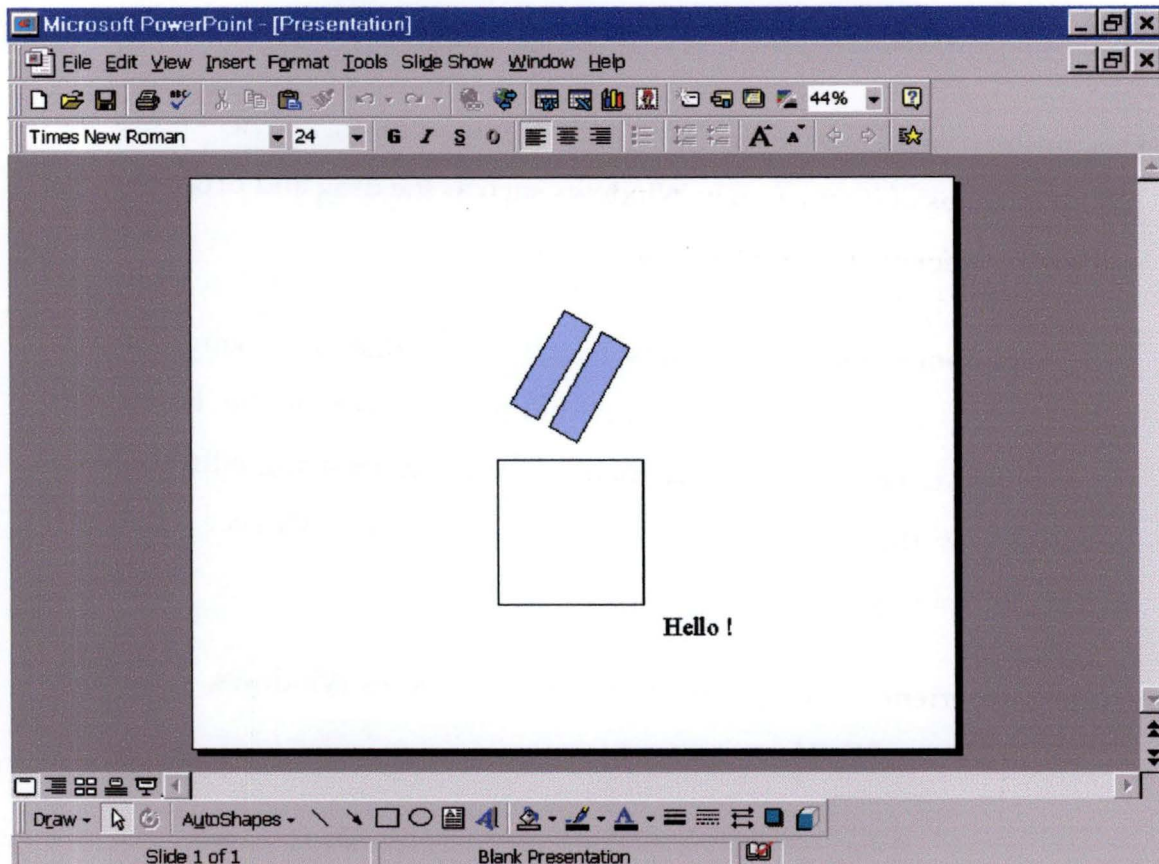


Figure 7 - CW example : Screenshot 2

## 2.2. The abstract task

There is no directly a real origin which corresponds to this task as it is partly an editing task. We won't develop here the abstract task here but the evaluator has to have in mind the structure of goals/subgoals thought by the user.

An analyse of the context of use of the interface

### 2.3. User profile

- **Description** : we assume the user is a Windows user, i.e. he knows the different means of interaction in Windows such as the drag and drop, clicking and double-clicking on the mouse and so on.
- **Task experience description** : this is the knowledge of a same kind of implemented task in a similar applications. We assume he hasn't this knowledge, i.e. he hasn't the experience of another drawing editor. This is made to put the conditions of a first-time user in evidence. The task experience is so « LOW ».
- **System experience description** : the future user knows Windows. The system experience is so « HIGH ».

### 2.4. Context of work

The context of work is an office workstation. As it is a generic interface, we don't suppose other assumptions about the environment of work.

### 2.5. Utility and usability criteria

We can assume the frequency of use may be high. So, the **learning time** may be high. The **rapidity of execution** hasn't necessary to be high. A minimal **error rate** isn't a critical criterion as an error will lead the user to find out other mechanisms of the interface and by the fact that the task hasn't a vital importance. It would be good if the **period of persistency** was enough high permitting the user to keep the aquired knowledge. The **subjective satisfaction** of the user is the most

important criterion we will retain. We recall the subjective satisfaction is the sensation of pleasure and comfort that the user perceives using the interface. For this kind of interface, we think it's very important as this kind of interface has to be as user-friendly as possible. The **degree of coverage** of the mechanisms of the interface has also to be high because the user wants to make any kind of drawing.

## 2.6. The concrete task (the implemented task)

Here is the decomposition in goal/subgoals of the task as it is imposed by the interface.

0. Main goal : Make the task of modification
1. Rotate the two rectangles of 60 degrees
  - 1.1. Group the two rectangles
    - 1.1.1. Drag a rectangle to select the rectangles
    - 1.1.2. Group the selected rectangles
  - 1.2. Rotate the selected grouped two rectangles of 60 degrees
    - 1.2.1. Select the Free Rotate icon in the drawing toolbar
    - 1.2.2. Rotate the selected grouped rectangles of 60 degrees
2. Draw a no-filled square below the rotated rectangles
  - 2.1. Select the rectangle icon in the drawing toolbar
  - 2.2. Draw a square below the two rotated rectangles
  - 2.3. Make the square no-filled
    - 2.3.1. Select the Fill color icon (the right part) in the drawing toolbar
    - 2.3.2. Select the No Fill MenuItem in the Fill Color menu
3. Put the word : « Hello ! » south-east of the square
  - 3.1. Select the Text Box icon in the drawing toolbar
  - 3.2. Put the word « Hello ! » south-east of the square

The first part of the task decomposition is finished. Now, it's time to make a more detailed description of each atomic subgoal to describe their actions.

**Method to achieve subgoal X : Select <object> icon in the drawing toolbar**

Comments : This method is parametrised to avoid describing repeatedly the same kind of method. This method is to select an icon in the drawing toolbar.

	User Action	System Feedback	System state
X1	~[object icon']	<ul style="list-style-type: none"> <li>• OBJECT ICON!</li> <li>• Wait(2sec): hint(object icon)</li> </ul>	
X2	M v ^	<ul style="list-style-type: none"> <li>• Object icon' !!</li> <li>• <math>\forall</math> object icon <math>\neq</math> object icon' : object icon-!!</li> <li>• ShowStatusBar(object icon')</li> <li>• The mouse pointer is evolving</li> </ul>	Selected = selected U {object icon'}

**Method to achieve subgoal 1.1.1. : Drag a rectangle to select the rectangles**

Main goal and active subgoal :

Main goal : Make the task of modification

Rotate the two rectangles of 60 degrees

Group the two rectangles

→ Drag a rectangle to select the rectangles

	User Action	System Feedback	System state
	'Determine the north-west of the two rectangles (x, y)'		
1	$\sim[x, y] M v$		
	'Determine the south-east of the two rectangles (x', y)''		
2	$\sim[x', y']$	Rectangle.ghost' >> ~	
3	$M ^$	<ul style="list-style-type: none"> <li>• Two rectangles' !</li> <li>• Erase (rectangle.ghost')</li> </ul>	Selected = selected U {2 rectangles}

### Method to achieve subgoal 1.1.2. : Group the selected rectangles

Main goal and active subgoal :

Main goal : Make the task of modification

Rotate the two rectangles of 60 degrees

Group the two rectangles

→ Group the selected rectangles

	User Action	System Feedback	System state
	'Determine a place above one of the two selected rectangle (x, y)'		

1	~[x, y]	The mouse pointer is evolving from a left-arrow pointer to a graphics repositioning pointer	
2	M v (right button) ^	<ul style="list-style-type: none"> <li>• A menu appears</li> <li>• The mouse pointer is evolving from a graphics repositioning pointer to a left-arrow pointer</li> </ul>	
	'Determine the MenuItem « Group » in the submenu « Grouping » (x', y)'		
3	~[x', y']	The different ItemMenu over-moused are colored in blue	
4	M v ^	The two rectangles are now selected and grouped as one shape	

**Method to achieve subgoal 1.2.1. Select the Free Rotate icon in the drawing toolbar**

Main goal and active subgoal :

Main goal : Make the task of modification

Rotate the two rectangles of 60 degrees

Rotate the grouped two rectangles

→ Select the Free Rotate icon in the drawing toolbar

Comments : we can use the parametrised method X to describe this method. Once this subgoal has been achieved, the handles around the selected grouped rectangles are changing from small white squares to small green circles. Moreover, the mouse pointer is evolving from a left-arrow pointer to a left-arrow one with an arrowed circle on its top.

The value of the hint function is : "Free Rotate"

**Method to achieve subgoal 1.2.2. Rotate the selected grouped rectangles of 60 degrees**

Main goal and active subgoal :

Main goal : Make the task of modification

    Rotate the two rectangles of 60 degrees

        Rotate the grouped two rectangles

            → Rotate the selected grouped rectangles of 60 degrees

	User Action	System Feedback	System state
	~[any rectangles.handle]	The pointer is evolving from left-arrow pointer with an arrowed circle on its top to an arrowed circle	
1	~[x, y] M v	The pointer is enlarging	
2	R 60°	Selected grouped	



		<code>rectangles.ghost' &gt;&gt; R</code>	
3	M ^	<ul style="list-style-type: none"> <li>• Erase (<code>rectangle.ghost'</code>)</li> <li>• The selected grouped rectangle appears rotated of 60 degrees</li> </ul>	

**Method to achieve subgoal 2.1. Select the rectangle icon in the drawing toolbar**

Main goal and active subgoal :

Main goal : Make the task of modification

Draw a no-filled square below the rotated rectangles

→ Select the rectangle icon in the drawing toolbar

Comments : we can use the parametrised method X to describe this method. The mouse pointer is evolving from a left-arrow pointer to a cross-hair pointer

The value of the hint function is : "Rectangle"

**Method to achieve subgoal 2.2. Draw a square below the two rotated rectangles**

Main goal and active subgoal :

Main goal : Make the task of modification

Draw a no-filled square below the rotated rectangles

→ Draw a square below the two rotated rectangles

	User Action	System Feedback	System state
	'Determine the upper left corner of the future square (x, y)'		
1	K v [Shift] ~[x, y] M v		
	'Determine the lower right corner of the future square (x', y)'		
2	~[x', y']	Square >> ~	
3	M ^ K ^[Shift]	<ul style="list-style-type: none"> <li>• Square' !</li> <li>• Square.color = green</li> </ul>	Selected = selected U {square}

**Method to achieve subgoal 2.3.1. Select the Fill color icon (the right part) in the drawing toolbar**

Main goal and active subgoal :

Main goal : Make the task of modification

Draw a no-filled square below the rotated rectangles

Make the square no-filled

→ Select the Fill color icon (the right part) in the drawing toolbar

Comments : we can use the parametrised method X to describe this method. Once this subgoal has been achieved, a menu with the different possible colors appears.

The value of the hint function is : "Fill Color"

**Method to achieve subgoal 2.3.2. Select the No Fill ItemMenu in the Fill Color menu**

Main goal and active subgoal :

Main goal : Make the task of modification

Draw a no-filled square below the rotated rectangles

Make the square no-filled

→ Select the No Fill ItemMenu in the Fill Color menu

	User Action	System Feedback	System state
1	~[No Fill MenuItem]	No Fill MenuItem !	Selected = selected U {No fill menuItem}
2	M v ^	Square.color = transparent	

**Method to achieve subgoal 3.1. Select the Text Box icon in the drawing toolbar**

Main goal and active subgoal :

Main goal : Make the task of modification

→ Put the word « Hello ! » south-east of the square

→ Select the Text Box icon in the drawing toolbar

Comments : we can use the parametrised method X to describe this method. The mouse pointer is evolving from a left-arrow pointer to a downward-pointing arrow.

The value of the hint function is : "Text Box"

**Method to achieve subgoal 3.2. Put the word « Hello ! » south-east of the square**

Main goal and active subgoal :

Main goal : Make the task of modification

→ Put the word « Hello ! » south-east of the square

→ Put the word « Hello ! » south-east of the square

	User Action	System Feedback	System state
	'Determine the downer right corner of the square (x, y)'		
1	~[x,y] M v ^	The mouse pointer is evolving from a downward-pointing arrow to a I-beam pointer	
2	K [Hello !]	Square.text = « Hello »	

### 3. Performing the Cognitive Walkthrough

#### *Analysis of subgoal 1 : Rotate the two rectangles of 60 degrees*

1. *Will the user try to achieve the right subgoal?*

Yes, by knowledge of the abstract task. This subgoal is present in the abstract and implemented task.

#### **Remark**

The order performing the subgoal has no importance. So, we assume the user will achieve their in the order we have define in the task description of the implemented task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

No, the user has no experience about this subgoal. But the interface can maybe help him to realize this subgoal.

#### *Analysis of subgoal 1.1. : Group the two rectangles*

1. *Will the user try to achieve the right subgoal?*

Probably not. Indeed, the user will try to rotate the two rectangles once they will be selected but won't necessary think about to group it. At this moment, the interface doesn't provide any prompt telling the user about rotating a set of selected shapes once they are selected.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

No, the user has no experience about this subgoal. But the interface can maybe help him to realize this subgoal.

#### ***Analysis of subgoal 1.1.1. : Drag a rectangle to select the rectangles***

1. *Will the user try to achieve the right subgoal?*

Yes, as he is a Windows user, he knows that he has to select a shape before performing actions (rotation) on it.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

The question has revealed that the user will know that he has to select the two rectangles. As the rubber-banding method the user will use here is present in Windows, for example to select a set of icons or files, the user has the knowledge to do this subgoal.

#### **Analysis of action 1 : $\sim[x, y] M v$**

3. *Can the user notice that the correct action is available?*

Probably because he has the experience of this kind of manipulation in Windows. Moreover, we can also justify by the answer to the question 2 of the current subgoal (inheritance).

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Probably because this action is also present in the mechanisms of Windows. For example, to select a set of files or icons. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

5. *Will the user perceive feedback?*

There is no feedback

6. *Will the user understand feedback?*

There is nothing to understand because there is no feedback.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

He will see progress in subgoal 1.1.1. and indirectly in subgoal 1.1., 1. and main goal. We can also justify that he will see progress by the answer to the question 2 of the current subgoal (inheritance).

#### **Analysis of action 2: ~ [x', y']**

3. *Can the user notice that the correct action is available?*

Yes, he knows this kind of manipulation in Windows. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

5. *Will the user perceive feedback?*

Yes, a ghost appears.

6. *Will the user understand feedback?*

Yes, this is an explicit feedback that the user knows. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

He will see progress in subgoal 1.1.1. and indirectly in subgoal 1.1., 1. and main goal. We can also justify that he will see progress by the answer to the question 2 of the current subgoal (inheritance).

### **Analysis of action 3 : M ^**

3. *Can the user notice that the correct action is available?*

Yes, he knows this kind of manipulation in Windows. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

5. *Will the user perceive feedback?*

Yes, the two rectangle are now selected.

6. *Will the user understand feedback?*

Yes, by experience of the system. We can also justify by the answer to the question 2 of the current subgoal (inheritance).

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will see that subgoal 1.1.1. is finished because he has experience of this kind of manipulation in Windows. He will see progress indirectly in



subgoal 1.1., 1. and main goal. We can also justify that he will see progress by the answer to the question 2 of the current subgoal (inheritance).

### *Analysis of subgoal 1.1.2. : Group the selected rectangles*

1. *Will the user try to achieve the right subgoal?*

Probably not. Indeed, the user will try to rotate the two rectangles once they will be selected but won't necessary think about to group it.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

If we assume that the user knows that he has to group the two rectangles, the user has to know the contextual menu which will permit him to group the objects. As he is a Windows user, he knows this possibility. We will know assess if he can realize this.

### **Analysis of action 1 : $\sim[x, y]$**

3. *Can the user notice that the correct action is available?*

Yes, by experience of moving the mouse. So, moving the mouse pointer over one of the two selected rectangles will be perceived by the user.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

As we assume he knows that he has to group the rectangles before rotating their, we can say that moving the mouse pointer over one of the two selected rectangles in a purpose to have a contextual menu will be maybe performed by the user. For example, he can think that the contextual menu will provide him some actions like grouping and depending on the context.

5. *Will the user perceive feedback?*

Yes, the stimulus-response feedback is high (when the mouse pointer moves). Furthermore, the mouse pointer is evolving from a left-arrow pointer to a graphics repositioning pointer.

6. *Will the user understand feedback?*

Maybe not because the change of the appearance of the pointer only tells to the user that he can move the selected rectangle.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will maybe see progress in subgoal 1.1.2. if, assuming he knows that he has to group the rectangles before rotating their, he thinks that the contextual menu will provide him some actions like grouping and depending on the context. Subgoal 1.1. , subgoal 1 and main goal remain unsatisfied.

### **Analysis of action 2 : M v (right button) ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating the mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

As we assume he knows that he has to group the rectangles before rotating their, we can say that clicking over one of the two selected rectangles in a purpose to have a contextual menu will be maybe performed by the user. For example, he can think that the contextual menu will provide him some actions like grouping and depending on the context.

5. *Will the user perceive feedback?*

Yes, he sees a menu where stands the MenuItem : « Grouping ».

6. *Will the user understand feedback?*

Yes.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will maybe see progress in subgoal 1.1.2. and 1.1. if, assuming he knows that he has to group the rectangles before rotating their, he will see progress in subgoal 1 and main goal.

### **Analysis of action 3 : ~[x', y']**

3. *Can the user notice that the correct action is available?*

Yes, by experience of moving the mouse towards menu and submenu.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

As we assume he knows that he has to group the rectangles before rotating their, he will search, he will associate this action with the subgoal he is trying to achieve.

5. *Will the user perceive feedback?*

Yes, the different MenuItem colour in blue when the mouse is over them.

6. *Will the user understand feedback?*

Yes, by experience of Windows.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will maybe see progress in subgoal 1.1.2. and 1.1. if, assuming he knows that he has to group the rectangles before rotating their, he will see progress in subgoal 1 and main goal.

#### **Analysis of action 4 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, because moving and clicking are often actions which have to be made one after the other.

5. *Will the user perceive feedback?*

Yes, the two rectangles are now selected as one shape.

6. *Will the user understand feedback?*

Yes, by experience of Windows.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will see that subgoal 1.1.2. and 1.1. are finished. If, assuming he knows that he has to group the rectangles before rotating their, he will see progress in subgoal 1 and main goal.

*Analysis of subgoal 1.2. : Rotate the selected grouped two rectangles of 60 degrees*

1. *Will the user try to achieve the right subgoal?*

Yes, because this subgoal is present in the abstract task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

A priori, he hasn't the knowledge to perform it as he has never done before.

We will assess if the system can help him performing guidance and prompts.

*Analysis of subgoal 1.2.1. : Select the Free Rotate icon in the drawing toolbar*

1. *Will the user try to achieve the right subgoal?*

Probably because the user knows that he has to made an action (rotation) on the selected and grouped rectangles. However, he doesn't maybe know that he has to select an icon to make a rotation and will maybe try to find out the possibility to rotate in the standard toolbar.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

He has the knowledge to select an icon because he has experience of the system.

**Analysis of action 1 : ~[Free Rotate]**

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

We delay our answer to question 6.

5. *Will the user perceive feedback?*

Yes, he can see « Free Rotate ».

6. *Will the user understand feedback?*

Yes and so, he will associate the correct action with the subgoal he is trying to achieve.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will see progress in subgoal 1.2.1., 1.2., 1 and main goal.

#### **Analysis of action 2 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, because moving and clicking are often actions which have to be made one after the other.

5. *Will the user perceive feedback?*

Yes, the handles around the selected grouped rectangles are changing from small white squares to small green circles. Moreover, the mouse pointer is evolving from a left-arrow pointer to a left-arrow one with an arrowed circle on its top.

6. *Will the user understand feedback?*

Probably, because the handles are now circle and especially because the mouse pointer is now a left-arrow with an arrowed circle on its top.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

Subgoal 1.2.1. is finished. User will see progress in subgoal 1.2., 1. and main goal.

***Analysis of subgoal 1.2.2. : Rotate the selected grouped rectangles of 60 degrees***

1. *Will the user try to achieve the right subgoal?*

Yes, because this subgoal is present in the abstract task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

A priori, he won't.

***Analysis of action 1 : ~[x, y] M v***

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably because the user has to rotate the selected grouped rectangles. So, he has to make an action on it. He will probably attempt this action in a purpose of performing an action on the selected grouped rectangles.

5. *Will the user perceive feedback?*

Yes, the mouse pointer is enlarging.

6. *Will the user understand feedback?*

Probably.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will see progress in subgoal 1.2.2., 1.2., 1 and main goal.

### **Analysis of action 2 : R 60°**

3. *Can the user notice that the correct action is available?*

He can see the rotation by reproducing with the mouse the action of rotating an object. However, it's difficult for the user to perceive that the rotation will be of 60 degrees.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably because the user has to rotate the selected grouped rectangles. So, to make an action of rotation with the mouse will permit the user to rotate the selected grouped rectangle. However, the subgoal he is trying to achieve is to make a rotation of 60 degrees.

In conclusion, we can say that he will associate the action of rotating with the subgoal he is trying to achieve but the action is very difficult to achieve as there is no feedback about the degrees (see question 5).

5. *Will the user perceive feedback?*

He can see a ghost of the selected grouped rectangle following the motion of the mouse but he can't see any information about the degree of rotation.



6. *Will the user understand feedback?*

Probably but this feedback is incomplete.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

User will see progress in rotation but as there is no information about the degree, he won't see a real progress towards the subgoal 1.2.2., 1.2., 1. and main goal.

### **Analysis of action 3 : M ^**

3. *Can the user notice that the correct action is available?*

Yes, as the mouse button is currently pressed and as he has experience of that in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, if we assume that he has succeeded to make a rotation of 60 degrees, the fact to release the mouse button is related to the accomplishment of the subgoal 1.2.2.

5. *Will the user perceive feedback?*

The ghost disappears and the selected grouped rectangle appears rotated of (maybe) 60 degrees.

6. *Will the user understand feedback?*

Yes.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

If we assume that he has succeeded to make a rotation of 60 degrees, the user will see that subgoal 1.2.2., 1.2. and 1. are finished. He will see progress in main goal which remains unsatisfied.

#### *Analysis of subgoal 2 : Draw a no-filled square below the rotated rectangles*

1. *Will the user try to achieve the right subgoal?*

Yes, by knowledge of the abstract task. This subgoal is present in the abstract and implemented task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

This will be assessed later when we will answer to questions about the subgoals and their actions of this current subgoal.

#### *Analysis of subgoal 2.1. : Select the rectangle icon in the drawing toolbar*

1. *Will the user try to achieve the right subgoal?*

Probably because the user knows that he has to make a square. However, he doesn't maybe know that he has to select this icon because this icon resembles to a rectangle and not to a square.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

He has the knowledge to select an icon because he has experience of the system.

### **Analysis of action 1 : ~[Rectangle]**

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

We delay our answer to question 6.

5. *Will the user perceive feedback?*

Yes, he can see « Rectangle ».

6. *Will the user understand feedback?*

Yes and so, he will maybe associate the correct action with the subgoal he is trying to achieve. We have said « maybe » because the subgoal is to draw a square, not a rectangle.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

If the user thinks that to draw a square, he has to select the rectangle icon, he will probably see progress in subgoal 2.1., 2. and main goal.

### **Analysis of action 2 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, because moving and clicking are often actions which have to be made one after the other.

5. *Will the user perceive feedback?*

Yes, the mouse pointer is evolving from a left-arrow pointer to a cross-hair one. The status bar indicates : « Click and drag to insert an AutoShape ».

6. *Will the user understand feedback?*

The cross-hair mouse pointer is enough explicit. The information given by the status bar won't maybe not understood by the user. This will be assessed in the next subgoal.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

If the user thinks that to draw a square, he has to select the rectangle icon, he will probably see that subgoal 2.1. is finished. He will see progress in subgoal 2. and main goal, which remain unsatisfied.

***Analysis of subgoal 2.2. : Draw a square below the two rotated rectangles***

1. *Will the user try to achieve the right subgoal?*

Yes, because this subgoal is present in the abstract task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

Maybe. This will be assess later.

**Analysis of action 1 : ~[x, y] M v and K v [Shift]**

3. *Can the user notice that the correct action is available?*

Probably not because no prompt is present to indicate that to draw a regular polygon (a square here), he has to press the « Shift » key, dragging the rectangle. (see question 5).

Moreover, as the feedback of the previous action is « Click and drag to insert an AutoShape », the user will probably try to click on the screen and only after that to drag a rectangle. A more appropriate feedback would have been : « Press and drag to insert a rectangle » instead of « Click and drag to insert an AutoShape ». Furthermore, this bad feedback maybe leads the user to click on the icon instead of clicking on the slide, as this bad feedback appears when the user clicks on the rectangle icon.

In conclusion, a good feedback would be : « Press and drag on the slide to insert a rectangle ».

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Even if we assume that the user knows that he has to press the « Shift » key and to drag the rectangle simultaneously, as he has no experience about this, he won't associate this action with the subgoal of drawing a square below the two rotated rectangles because there is no feedback about this (see question 5).

5. *Will the user perceive feedback?*

There is no feedback

6. *Will the user understand feedback?*

There is nothing to understand because there is no feedback.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

Even if we assume that the user knows that he has to press the « Shift » key and to drag the rectangle simultaneously, the user won't see directly a progress.

### **Analysis of action 2 : ~ [x', y']**

3. *Can the user notice that the correct action is available?*

Yes, he knows this kind of manipulation in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably.

5. *Will the user perceive feedback?*

Yes, the square appears and is enlarging.

6. *Will the user understand feedback?*

Yes, this is an explicit feedback.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will see progress in subgoal 2.2., 2. and main goal.

### **Analysis of action 3 : M ^ and K ^[Shift]**

3. *Can the user notice that the correct action is available?*

Yes, he knows this kind of manipulation in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably.

5. *Will the user perceive feedback?*

Yes, the rectangle (square) is selected and is filling with the color green.

6. *Will the user understand feedback?*

Yes but the square is filling with the color green. So, there is a lack of consistency between the filling of the icon, which is transparent and the result on the slide.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

Subgoal 2.2. is finished. The user will probably see progress in subgoal 2. and main goal.

### ***Analysis of subgoal 2.3. : Make the square no-filled***

1. *Will the user try to achieve the right subgoal?*

Yes, because this subgoal is present in the abstract task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

Maybe. This will be assess later.

*Analysis of subgoal 2.3.1. : Select the Fill Color icon (the right part) in the drawing toolbar*

1. *Will the user try to achieve the right subgoal?*

Probably because the user knows that he has to fill in the square. However, he doesn't maybe know that he has to select the right part of this icon.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

He has the knowledge to select an icon because he has experience of the system.

**Analysis of action 1 : ~[Fill Color (right part)]**

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

We delay our answer to question 6.

5. *Will the user perceive feedback?*

Yes, he can see « Fill Color ».

6. *Will the user understand feedback?*

The two parts of this icon have the same feedback « Fill Color ». The user may be confused by this ambiguity.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*



He will probably see progress in subgoal 2.3.1, 2.3., 2. and main goal.

### **Analysis of action 2 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, because moving and clicking are often actions which have to be made one after the other.

5. *Will the user perceive feedback?*

Yes, a menu appears once this subgoal is achieved. This menu shows the different colors of filling.

6. *Will the user understand feedback?*

Yes, by experience of Windows.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

Subgoal 2.3.1. is finished. He will probably see progress in subgoal 2.3., 2. and main goal.

### **Analysis of subgoal 2.3.2. : Select the No Fill MenuItem in the Fill Color menu**

1. *Will the user try to achieve the right subgoal?*

Probably because the user knows that he has to fill in the square. However, he doesn't maybe know that he has to select the right part of this icon.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

He has the knowledge to navigate towards menu.

#### **Analysis of action 1 : ~[No Fill MenuItem]**

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes.

5. *Will the user perceive feedback?*

The No Fill MenuItem is highlighted.

6. *Will the user understand feedback?*

Yes, by experience of the system.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

He will probably see progress in subgoal 2.3.2., 2.3., 2. and main goal.

#### **Analysis of action 2 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*  
Yes, because moving and clicking are often actions which have to be made one after the other.
5. *Will the user perceive feedback?*  
Yes, the square is transparent.
6. *Will the user understand feedback?*  
Yes.
7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*  
Subgoal 2.3.2., 2.3. and 2 are finished. The user will see progress in main goal.

***Analysis of subgoal 3. : Put the word: "Hello !" south-east of the square***

1. *Will the user try to achieve the right subgoal?*  
Yes, because this subgoal is present in the abstract task.
2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*  
This will be assess later.

***Analysis of subgoal 3.1. : Select the Text Box icon in the drawing toolbar***

1. *Will the user try to achieve the right subgoal?*

Probably.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

He has the knowledge to select an icon because he has experience of the system.

### **Analysis of action 1 : ~[Text Box]**

3. *Can the user notice that the correct action is available?*

Yes, he can perceive it, by experience of manipulating a mouse in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

We delay our answer to question 6.

5. *Will the user perceive feedback?*

Yes, he can see « Text Box ».

6. *Will the user understand feedback?*

Yes and so, he will associate the correct action with the subgoal he is trying to achieve.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will probably see progress in subgoal 3.1. and indirectly in subgoal 3. and main goal.

## **Analysis of action 2 : M v ^**

3. *Can the user notice that the correct action is available?*

Yes, by experience of manipulating a mouse.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, because moving and clicking are often actions which have to be made one after the other.

5. *Will the user perceive feedback?*

Yes, the mouse pointer is evolving from a left-arrow pointer to a downward-pointing arrow. The status bar indicates: « Click and drag to insert a TextBox ».

6. *Will the user understand feedback?*

The downward pointing arrow pointer isn't enough explicit. The information given by the status bar: « Click and drag to insert a TextBox » won't maybe not understood by the user. This will be assessed in the next subgoal.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will probably see progree that subgoal 3.1. is finished. The user will probably see indirectly progress in subgoal 3. and main goal.

## **Analysis of subgoal 3.2. : Put the word "Hello !" south-east of the square**

1. *Will the user try to achieve the right subgoal?*

Yes, because this subgoal is present in the abstract task.

2. *What knowledge is needed to achieve the right subgoal? Will the user have this knowledge?*

This will be assessed later.

### **Analysis of action 1 : $\sim[x, y] M v \wedge$**

3. *Can the user notice that the correct action is available?*

Perhaps because as the feedback of the previous action is « Click and drag to insert a Text Box », the user will probably try to click on the screen and only after that to drag a Text Box. A more appropriate feedback would have been : « Press and drag to insert a Text Box » instead of « Click and drag to insert a Text Box ». Furthermore, this bad feedback maybe leads the user to click on the icon instead of clicking on the slide, as this bad feedback appears when the user clicks on the rectangle icon.

In conclusion, a good feedback would be : « Press and drag on the slide to insert a Text Box ». This is the same kind of problem we met inserting a square on the slide (see subgoal 2.2.)

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Probably.

5. *Will the user perceive feedback?*

The mouse pointer is evolving from a downward pointing arrow to a I-beam pointer.

6. *Will the user understand feedback?*

This is an explicit feedback.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The user will maybe see progress in subgoal 3.2., 3. and main goal.

### **Analysis of action 2 : K [Hello !]**

3. *Can the user notice that the correct action is available?*

Yes, he knows this kind of manipulation in Windows.

4. *Will the user associate the correct action with the subgoal they are trying to achieve?*

Yes, probably.

5. *Will the user perceive feedback?*

Yes, the word is appearing.

6. *Will the user understand feedback?*

Yes, this is an explicit feedback.

7. *Will the user see that progress is being made towards solution of their task in relation to their main goal and current subgoals?*

The subgoal 3.2., 3. and the main goal are finished.

## **4. Results of the method and their interpretation**

### **Problem 1 : The grouping of objects before the rotation**

#### *Description of the problem*

The user will probably select the two rectangles before rotating them but won't necessary think about to group them before.

#### *Interaction point*

Subgoal 1.1. and 1.1.2., CW1

#### *Likely concrete user difficulties in context*

If the user will think about this problem, he will find out that the result obtained (screenshot 3) is logical. Indeed, after selecting the two rectangles, if the user rotates them, they will rotate each separately. The fact of selecting two objects means that the future action will be occur on each one separately but in the same time.

The user will maybe think about that and will maybe try to reach the contextual menu which will permit him to group the two rectangles and to rotate them after (screenshot 4).



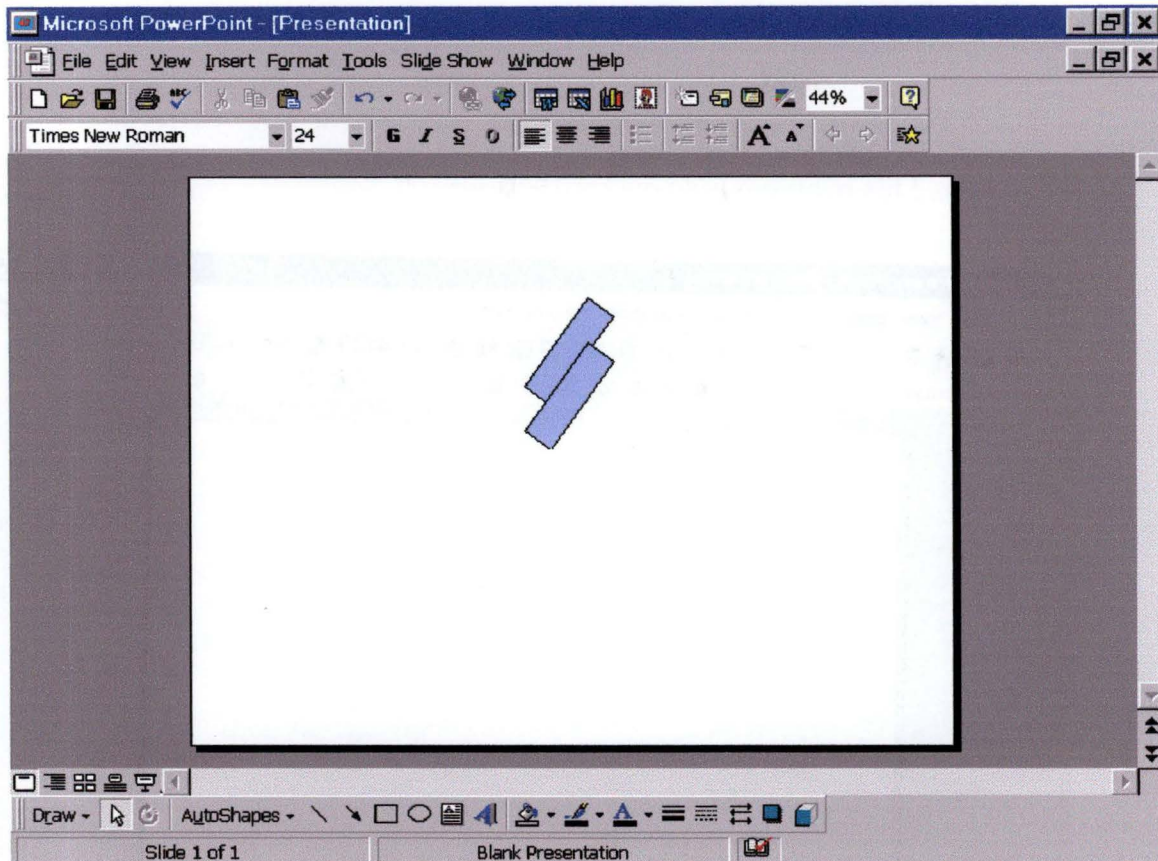


Figure 8 - CW example : Screenshot 3

### *The assumed causes of these difficulties*

The causes are not necessary related to a mistake of the interface. If the user thinks about this manipulation, we will maybe find what's wrong with this. However, the interface could provide a kind of guidance, after the two rectangles are selected and after the Free Rotate icon is selected, telling the user that to rotate the two rectangles considered as one shape, he has to group them first.

### *The success/failure case*

The problem will eventually prevent the user performing correctly the entire task. This problem is so enough serious.

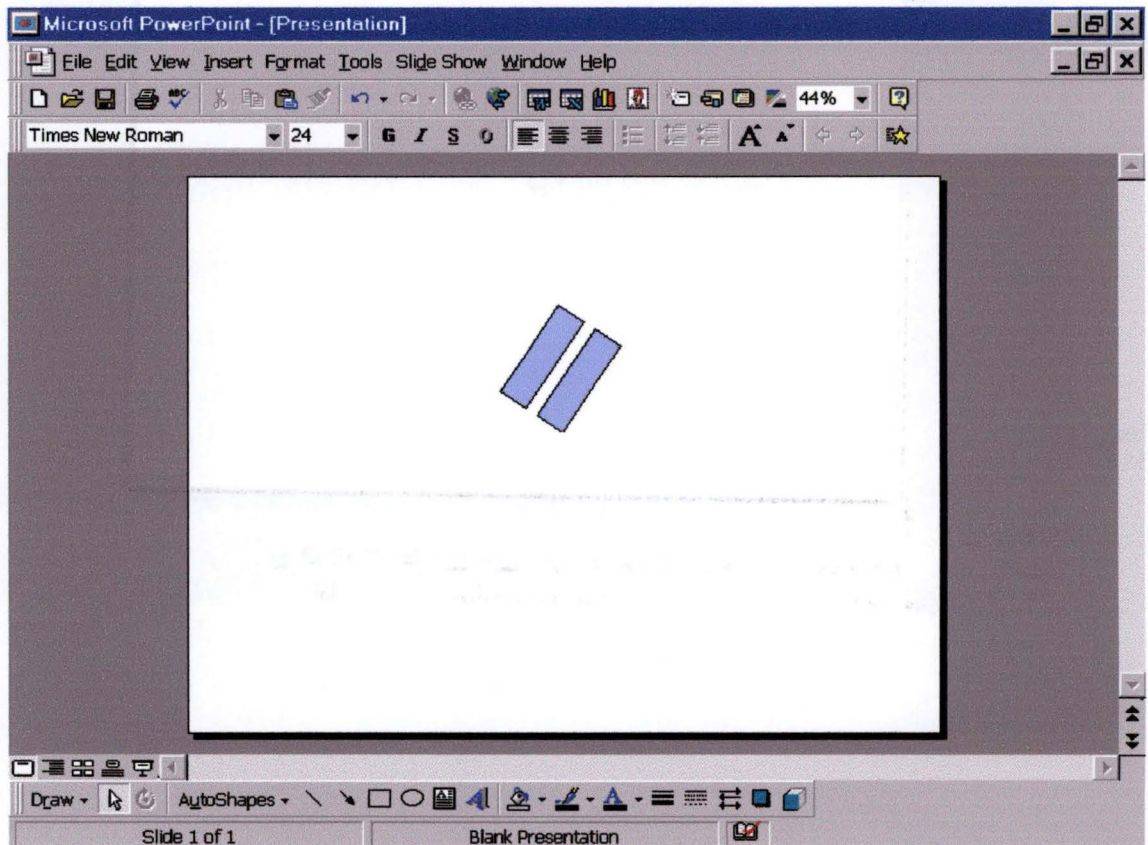


Figure 9 - CW example : Screenshot 4

### *The seriousness of the problem*

The problem leads the user to waste some time but the time spend will permit him to learn the interface. The error rate is not a criteria on which we have put a big importance. In fact, as we have put a big importance on the subjective satisfaction, the problem is enough serious because this criteria comes into play.

*Concrete interactive object concerned by the problem*

The guidance associated to the Free Rotate icon which doesn't exist.

*A design suggestion to solve the problem*

The interface could provide a kind of guidance, after the two rectangles are selected and after the Free Rotate icon is selected, telling the user that to rotate the two rectangles considered as one shape, he has to group them first.

## **Problem 2 : The rotation through an angle of 60 degrees**

*Description of the problem*

After selecting and grouping the two rectangles, the user has to rotate them by 60 degrees. Any information is provided for the user to accomplish a such accurate rotation.

*Interaction point*

Subgoal 1.2., 1.1.2., action 2, CW5

*Likely concrete user difficulties in context*

The user doesn't know when he has to stop the rotation. So, he will probably try to make an approximation of 60 degrees.

*The assumed causes of these difficulties*

There is no feedback about the degrees of the current rotation (screenshot 5).

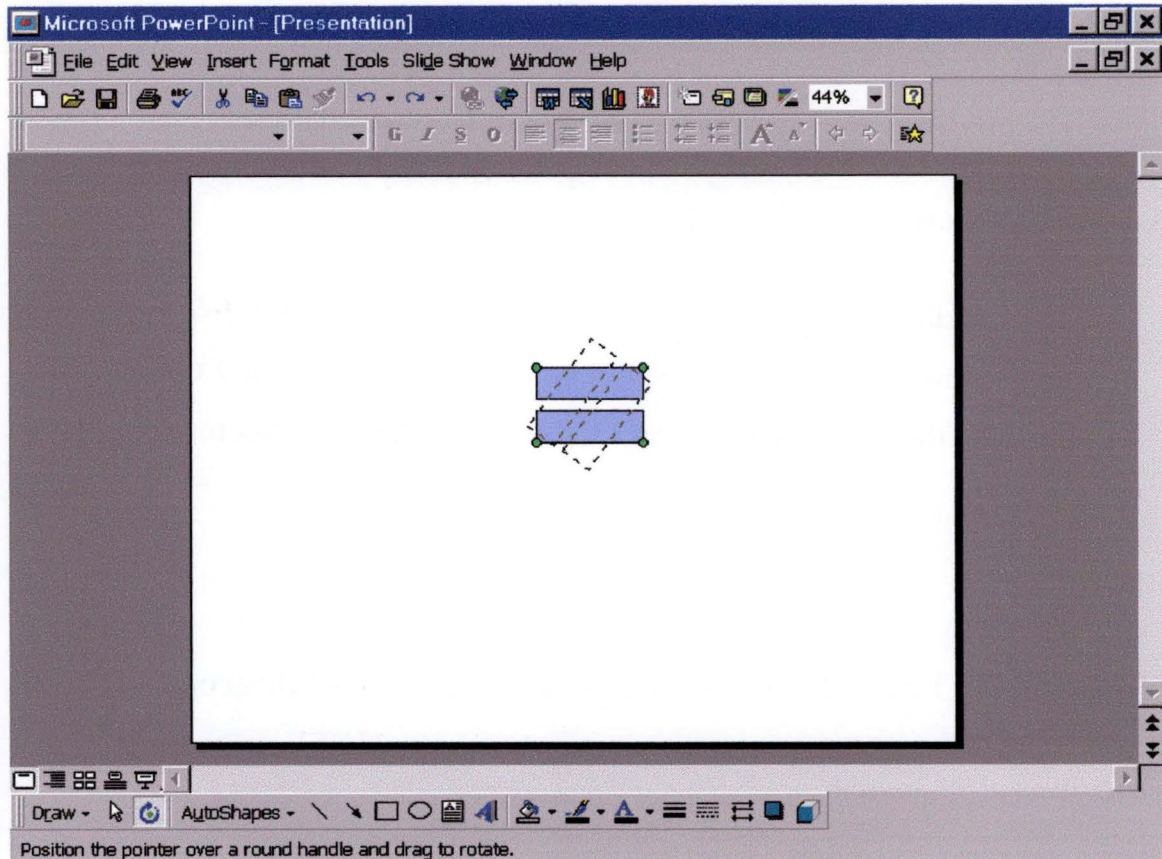


Figure 10 - CW example : Screenshot 5

### *The success/failure case*

This problem doesn't probably lead the user to a failure case because he will probably try to make an approximation of 60 degrees.

### *The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a negative answer to a question (question 5 here) and a big importance to the subjective satisfaction of the user means that the problem is serious.

Indeed, the lack of feedback doesn't bring a sensation of pleasure and comfort to the user using the interface.

Let's notice however that we have a feedback during the rotation. It's the ghost of the selected grouped rectangles. This is a good feedback but another feedback (the information feedback about the degrees of the current rotation) would have been necessary.

*Concrete interactive object concerned by the problem*

The guidance associated to the Free Rotate icon which doesn't exist.

*A design suggestion to solve the problem*

An information feedback about the degrees for the current rotation has to be inserted in the interface. To this aim, the designer has two solutions :

First, he can insert it in the status bar (screenshot 6). This solution is usual in Windows application. The drawback of this solution is that the user's attention is not focalized on this part of the screen (the status bar). Indeed, the user's attention is focalized where the current rotation is realized, on the ghost of the two selected and grouped rectangles rotating.

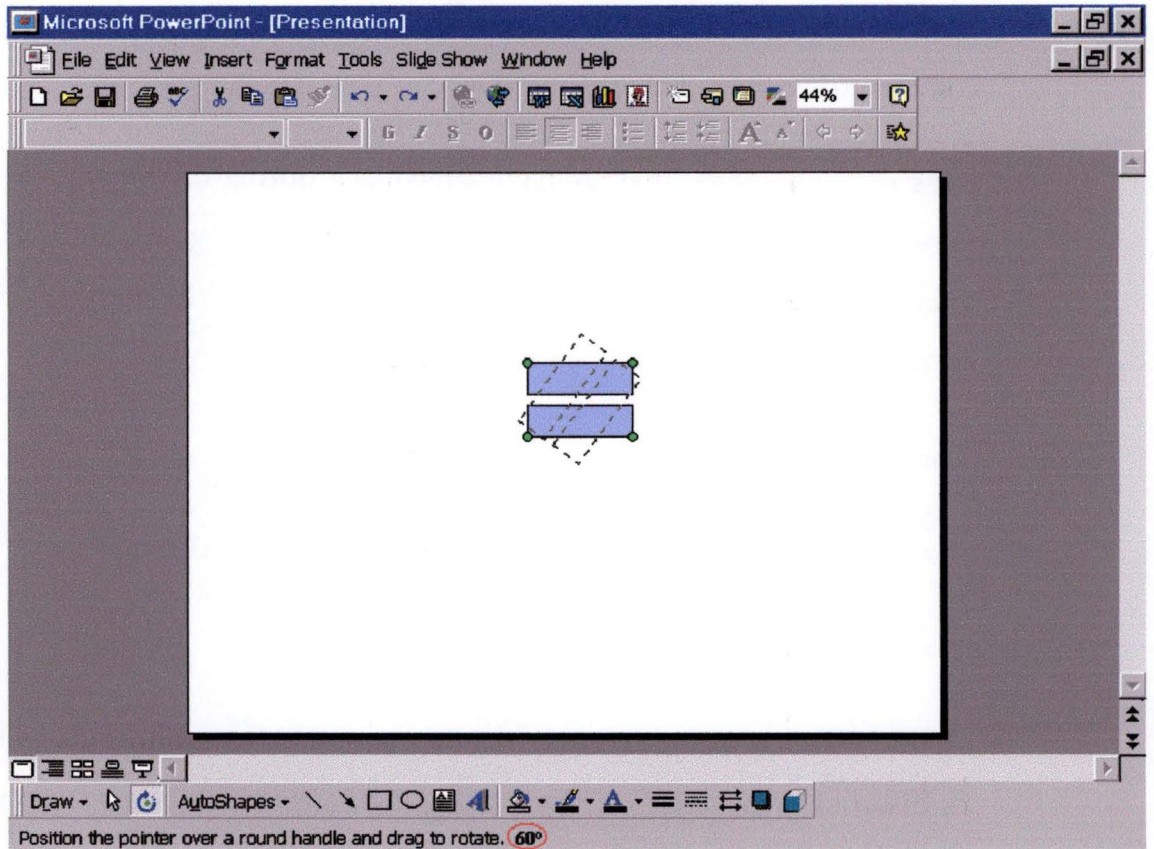


Figure 11 - CW example : Screenshot 6

The other solution is to display the current degrees near the user's attention. This solution permits the user to have in his sight both the ghost of the two selected and grouped rectangles rotating and the current degree of the rotation (screenshot 7).

Finally, let's notice for the story that PowerPoint 4.0, which is a previous version of which we made this application of the method, includes the degrees of rotation in the status bar !!!

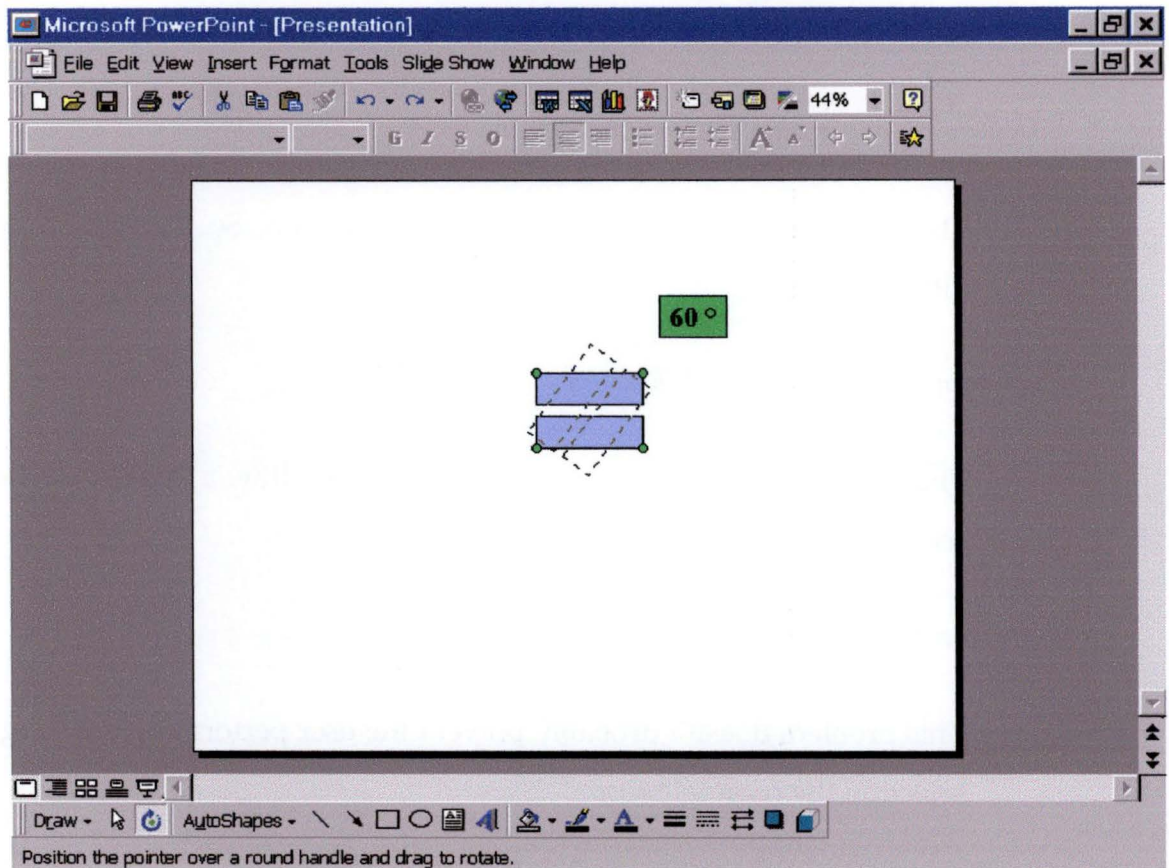


Figure 12 - CW example : Screenshot 7

### **Problem 3 : The drawing of a square (first problem)**

#### *Description of the problem*

The user has to draw a square. To do this, he has to select the rectangle icon. There is a lack of consistency by the fact that to draw a **square**, we have to select the **rectangle** icon.

#### *Interaction point*

## Subgoal 2.1., CW1

### *Likely concrete user difficulties in context*

The user will probably select the rectangle icon. So, there is no a real difficulty here.

### *The assumed causes of these difficulties*

There is a lack of consistency by the fact that to draw a **square**, we have to select the **rectangle** icon.

### *The success/failure case*

This problem doesn't probably prevent the user performing the subgoal of selecting the rectangle icon.

### *The seriousness of the problem*

This problem doesn't really go against the balance of the criteria we have provided.

### *Concrete interactive object concerned by the problem*

The rectangle icon.

### *A design suggestion to solve the problem*

It would be good if the drawing toolbar can include both the rectangle and the square icon even if we know that a square is a rectangle and moreover, experienced users know that to draw a regular shape, they have to press the « Shift » key dragging the shape.



So, another solution would have been to give a prompt to user telling him that to draw a square, he has to select the rectangle icon and press the « Shift key » dragging his shape. But this solution has the disadvantage of when to display this information ? If we say that this information has to be provided when the mouse pointer is over the rectangle icon or when the user select this icon, that means the user will have tried to do something whith this icon, the rectangle icon. So, the problem is not resolved.

In conclusion, we can say that, as the rectangle icon is the closest shape near the square and as the square is a rectangle, the user will probably select it even he dosen't know yet how he will draw an accurate square.

#### **Problem 4 : The drawing of a square (second problem)**

##### *Description of the problem*

When the user selects the rectangle icon, a feedback is provided : « Click and drag to insert an AutoShape ». This maybe leads the user to difficulties.

##### *Interaction point*

Subgoal 2.1., action 2, CW6

##### *Likely concrete user difficulties in context*

As, the feedback of the action is « Click and drag to insert an AutoShape », the user will probably try to click on the screen and only after that to drag a rectangle. A more appropriate feedback woul have been : « Press and drag to insert a rectangle » instead of « Click and drag to insert an AutoShape ». Furthermore, this bad feedback maybe leads the user to click

on the icon instead of clicking on the slide, as this bad feedback appears when the user clicks on the rectangle icon.

Finally, let's notice however that the user will maybe perform the right action by experience of dragging box in the Windows environment.

#### *The assumed causes of these difficulties*

There is a problem of vocabulary. A good feedback would be : « Press and drag on the slide to insert a rectangle ».

#### *The success/failure case*

This problem doesn't probably prevent the user performing this action but maybe leads him to encounter difficulties for the next subgoal.

#### *The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a negative answer to a question (question 6 here) and a big importance to the subjective satisfaction of the user means that the problem is serious.

Indeed, A problem of vocabulary doesn't bring a sensation of pleasure and comfort to the user using the interface.

Let's notice however that the user will maybe not see the information in the status bar because it's not in the user's attention but this is not a good argument as this bar is designed to help user.

#### *Concrete interactive object concerned by the problem*

The status bar.

*A design suggestion to solve the problem*

A good feedback in the status bar would be : « Press and drag on the slide to insert a rectangle ».

## **Problem 5 : The drawing of a square (third problem)**

*Description of the problem*

The user doesn't know that he has to press the « Shift » key dragging his rectangle.

Moreover, as we have seen in the previous problem, the bad feedback provided by the status bar maybe leads him to difficulties. We don't consider any more these difficulties (see previous problem for more explanations). We concentrate here on the « Shift » key problem.

*Interaction point*

Subgoal 2.2., action 1, CW3

*Likely concrete user difficulties in context*

The user won't probably try to press the shift key, dragging his rectangle. So, he will probably approximate a square.

*The assumed causes of these difficulties*

There is no information feedback, after selecting a rectangle, which will permit to help users drawing a square.

*The success/failure case*

This problem doesn't probably lead the user to a failure case because he will probably try to make an approximation of a square.

#### *The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a negative answer to a question (question 3 here) and a big importance to the subjective satisfaction of the user means that the problem is serious.

Indeed, A lack of guidance doesn't bring a sensation of pleasure and comfort to the user using the interface.

#### *Concrete interactive object concerned by the problem*

The status bar.

#### *A design suggestion to solve the problem*

A good feedback in the status bar would be : « Press and drag on the slide to insert a rectangle » as we have already said. Now, we complete this feedback saying : « Press and drag on the slide to insert a rectangle ; Press 'Shift' key to insert a square ». Let's notice that in PowerPoint 4.0, there is an information about the use of the « Shift » key when the icon is selected !!!

## Problem 6 : The filling of the shape

### *Description of the problem*

The square is filled with the colour green. The icon has a transparent filling (screenshot 8).

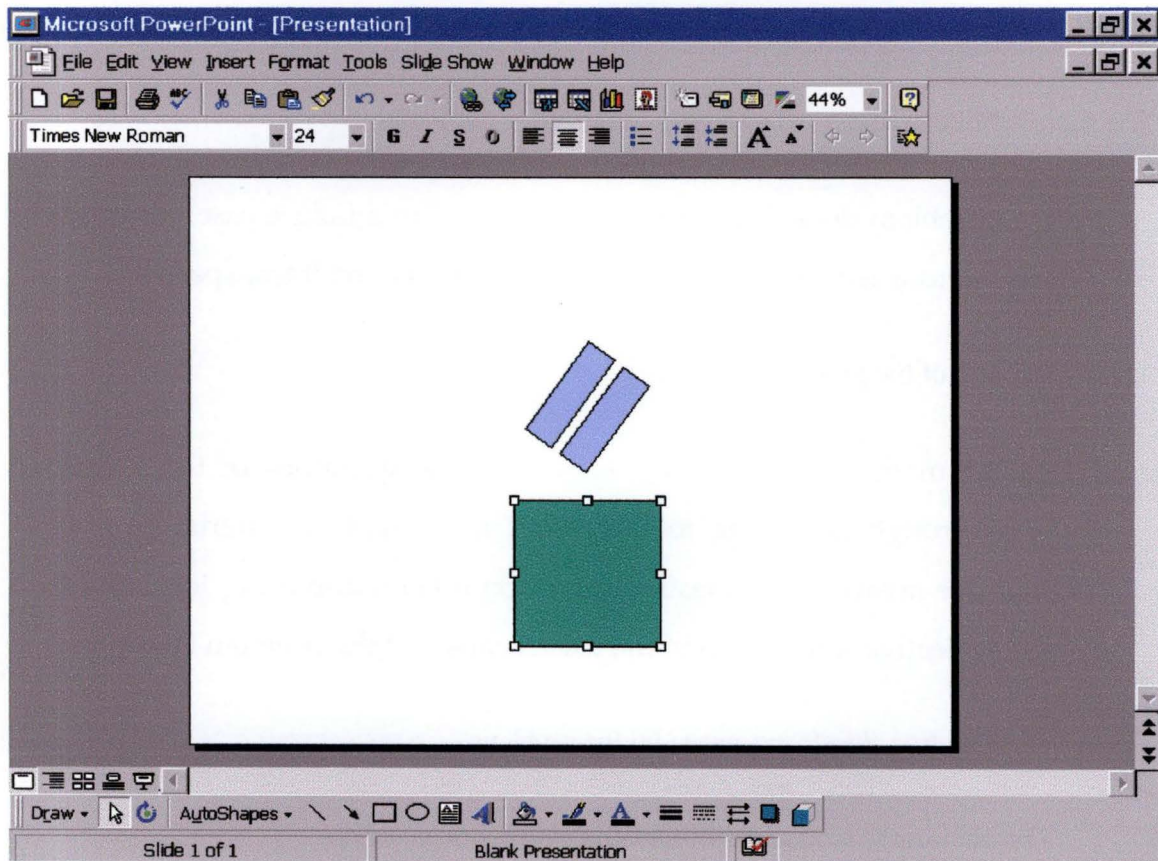


Figure 13 - CW example : Screenshot 8

### *Interaction point*

Subgoal 2.2., action 3, CW6

*Likely concrete user difficulties in context*

A waste of time to understand what is happening and the time putting another filling.

*The assumed causes of these difficulties*

A lack of consistency.

*The success/failure case*

This problem doesn't probably lead the user to a failure case but this will impose to waste time re-filling the shape (or making it transparent).

*The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a negative answer to a question (question 6 here) and a big importance to the subjective satisfaction of the user means that the problem is serious.

*Concrete interactive object concerned by the problem*

The rectangle icon (and in fact the other shape icons)

*A design suggestion to solve the problem*

When the icon is selected, the shape will have to be transparent.

## **Problem 7 : Selecting the right part of the Fill Color icon**

*Description of the problem*

The user will probably know that he has to fill in the square (with transparent colour, i.e. no filling) and so that he has to select this icon. However, he doesn't maybe know that he has to select the right part of this icon.

*Interaction point*

Subgoal 2.3.1., CW1 and action 1, CW6

*Likely concrete user difficulties in context*

A waste of time, for example trying to select the left part of this icon.

*The assumed causes of these difficulties*

There is no a real problem in the interface but there is the same feedback when the mouse is over both the left and right part of this icon.

*The success/failure case*

This problem doesn't probably lead the user to a failure case but this will impose to waste time.

*The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a negative answer to a question (question 6 here) and a big importance to the subjective satisfaction of the user means that the problem is serious.

*Concrete interactive object concerned by the problem*

The Fill Color icon

*A design suggestion to solve the problem*

Two feedbacks, according the part of the icon will lead to an unambiguous situation.

## **Problem 8 : The Text Box mouse pointer**

*Description of the problem*

When the user has selected the Text Box icon, the mouse pointer is evolving from a left-arrow pointer to a downward-pointing arrow. The user would probably expect an I-beam pointer.

Moreover, there is the same problem we have already encountered about the feedback of selecting a rectangle icon. The feedback here is a bit different: « Click and drag to insert a text box ». We can say the same remarks we have made here.

*Interaction point*

Subgoal 3.1., action 2, CW6

*Likely concrete user difficulties in context*

The user will maybe wonder if he has selected the right icon because he doesn't identify the expected I-beam pointer which is often present in the environment of Windows.



*The assumed causes of these difficulties*

This is not the usual mouse pointer.

*The success/failure case*

This problem doesn't probably lead the user to a failure case but this will impose to waste time.

*The seriousness of the problem*

As the matrix, related to the answers to the questions of the Cognitive Walkthrough according to the utility and usability criteria, tells us, a mitigated answer to the question 6 and a big importance to the subjective satisfaction of the user means that the problem has to be take in consideration.

*Concrete interactive object concerned by the problem*

The mouse pointer

*A design suggestion to solve the problem*

To change the downward-pointing arrow pointer to a I-beam pointer.

## 5. Conclusion

We have found very focused problems about this interface. Some of those problems are important.

Moreover, we can sometimes generalize one problem to other problems of the same kind. For example, the feedback associated of the selection of a shape is not enough explicit and we have already seen two places where it occurs: the rectangle icon and the text box one. We can easily generalize this problem to other icons.

Finally, we can find problems simply by manipulating the interface. For example, we have found that when we have rotated the two selected and grouped rectangles of a certain degree, it's sometimes impossible to go back to the initial position (one of the two rectangles is a bit distorted). Perhaps this problem can be generalize to any shape.

## *Appendix C*

### SPECIFICATION OF THE COGNITIVE WALKTHROUGH EDITOR

## **1. Introduction**

This appendix is aimed to give the data conceptual model of the Information System and the treatments conceptual model of it. These two documents are provided to complete the third part of our thesis.

## **2. Data conceptual model of the Information System**

### **2.1. ERA model of the Information System**

Here is the ERA model of the Information System (Fig. 14). It provides all the data considered and their relations.



## 2.2. Data dictionary

### USER PROFILE

This type of entity is associated to the description of the user considered in one cognitive walkthrough.

- **DESCRIPTION** : this attribute explains in large letters the profile of the user the analyst has to take in consideration.
- **TASK\_EXP** : this attribute is used to evaluate the considered task experience of the considered type of user. Three types of values are available : LOW, AVERAGE and HIGH.
- **TASK\_EXP\_DESC** : this attribute includes a complementary explanation of the task experience level.
- **SYS\_EXP** : this attribute gives the estimated value of the user experience level in the system where the task must be performed. Three types of values are available : LOW, AVERAGE and HIGH.
- **SYS\_EXP\_DESC** : No mandatory description of the user system experience.

### CONTEXT OF WORK

This type of entity is associated to the description of the context of work in one cognitive walkthrough.

- **DESCRIPTION** : This attribute explains in large letters the context of work of the user the analyst has to take in consideration.
- **VISIBILITY** : this attributes gives the estimated value of the brightness and the size in which the user will perform the task. Three types of values are available : POOR, AVERAGE and GOOD.
- **NOISE** : this attribute gives the estimated value of the noise in which the user has to perform the task. Three types of values are available : QUIET, AVERAGE, NOISY.

- **TREATMENT ALLOCATION** : this attribute can take two values : « SINGLE » if during the task performing on the interface by the user, anything is made. « MULTI » if during the task performing on the interface by the user, another thing(s) is (are) made.

### **CRITERIA**

This type of entity indicates the name and the balance of each of the utility and usability criteria.

- **NAME** : this attribute indicates the name of the six utility and usability criteria : LEARNING TIME, RAPIDITY OF EXECUTION, ERROR RATE, PERSISTENCY, SATISFACTION OF THE USER, COVERAGE.
- **BALANCE** : this attribute gives the estimated value of the importance of each of the utility and usability criteria. Five values are available :LOW, AVERAGE, HIGH.

### **TASK**

This type of entity is associated with the description of the considered task in the cognitive walkthrough.

- **DESCRIPTION** : this attribute is related to the description of the considered task.

### **SUBGOAL**

This type of entity describes the different goals/subgoals which compose the task description.

- **NUM** : this attribute identifies the type of entity Subgoal.
- **NAME** : this gives the name of the subgoal.
- **DESCRIPTION** : this attribute is a text description of the subgoal.

- **ATOMIC** : this boolean indicates if the subgoal can be further decomposed into lower subgoals or can't.

### **ACTION**

This type of entity describes the different actions which compose an atomic subgoal.

- **NUM** : this is an identifier of the action of a given subgoal.
- **VALUE** : this represents in UAN or in informal language the action.
- **SYSTEM FEEDBACK** : this represents in UAN or in informal language the feedback.
- **SYSTEM STATE** this represents in UAN or in informal language the system state.

### **ANSWER**

This type of entity describes the answers of the Cognitive Walkthrough and their value

- **DESCRIPTION** : this attribute gives the answer description given for a certain subgoal or action to a certain question.
- **VALUE** : the three values available are : YES, NOT or PERHAPS.

### **PROBLEM**

This type of entity describes a problem revealed by the method, their location, their seriousness.

- **PROBLEM NUMBER** : this attribute indicates the number of the problem in order they were found.
- **DESCRIPTION** : this attribute describes the problem.
- **INTERACTION POINT** : this attribute indicates the location of the problem in the hierarchy of goal/subgoals/ actions.

- **CONCRETE DIFFICULTIES** : this attribute explains the concrete difficulties of the user dealing with the problem.
- **CAUSE** : this attribute explains the cause(s) of the problem.
- **SERIOUSNESS** : according to the balance allowed to the utility and usability criteria, this attribute is trying to put a level of seriousness on the problem to find out the failure case. This attribute can take the following values : LOW, AVERAGE and HIGH.
- **CIO (CONCRETE INTERACTIVE OBJECT) CONCERNED** : this attribute indicates the piece of the interface which poses a problem



### 3. Treatments conceptual model of the IS

#### 3.1. Inputs specification

##### Objectives

To pick up data related to the inputs of the Cognitive Walkthrough which will permit to balance the evaluation of the interface with the method : user profile (his different levels of system and task knowledge), the context of work (its different levels of visibility, noise and treatment allocation) and the utility and usability criteria (the balance of each of the criteria).

##### Input messages

Input =

$\text{tool} \wedge \text{user\_profile} \wedge \text{context\_of\_work} \wedge \text{criteria} \wedge \text{task}$

Tool =

$(\text{name\_tool} \wedge \text{interface version})$

The description of the tool from which we want to evaluate the interface is limited to its identification and its version number (to differentiate the successive use of the method on different versions of the interface for a given task).

User\_profile =

$(\text{description} \wedge \text{task\_exp} \wedge \text{task\_desc} \wedge \text{sys\_exp} \wedge \text{sys\_desc})$

The user profile is firstly defined by a description of the kind of users considered and their motivations in the realization of the task assigned to them. The balance related to their knowledge of the task and the system on which this task must be performed is also to be provided.

Context\_of\_work =

$(\text{description} \wedge \text{visibility} \wedge \text{noise} \wedge \text{treatment allocation})$

The context of work is firstly defined by a description of the context of work considered. The balance related to the visibility of the environment of work, the noise of a such environment and the treatment allocation is also to be provided.

Criteria =

(Learning\_time  $\wedge$  Exec\_rapidity  $\wedge$  Error\_rate  $\wedge$  Persistency  $\wedge$  Satisfaction  $\wedge$  Coverage)

At each utility and usability criteria, a weight must be associated, this will permit to create a knowledge base to find out where problems may occur in function of the answers given to the questions.

Task =

(description)

There is a brief description of the considered task.

### **Output message**

Error\_mess

This error message is sent if the method has already be performed on the considered task, with the same version of the interface and with the same user profile. The message must also indicate that the consultation of the result of the already performed method are available.

### **Actions on the IS**

Creation of a new instance of the Cognitive Walkthrough

### **Treatment rules**

At the creation of a new instance to perform the method, the fact that the method has not previously be realized on the same task with the same tool must be verified.

## **3.2. Knowledge base generation**

### **Objectives**

From the utility and usability criteria, a knowledge base must be created. According to the answers given to the questions, we will then refer to this knowledge base to see if there may be problems in the interface.

### **Input messages**

Criteria =

Learning\_time  $\wedge$  Exec\_rapidity  $\wedge$  Error\_rate  $\wedge$  Persistency  $\wedge$  Satisfaction  $\wedge$   
Coverage

At each utility and usability criteria, a weight must be associated what will permit to create a knowledge base to find out where light and serious problems may occur in function of the answers given to the questions.

#### **Actions on the IS**

The different balance of the criteria must be recorded.

#### **Treatment rules**

None

### **3.3. Usual answers generation**

#### **Objectives**

According to the users' profile and the context of work, the different usual answers for each question will be generated, in a view to offer an intelligent tool to perform Cognitive Walkthrough.

#### **Input messages**

Task\_level

This indicates the level of knowledge that the considered user has of the task.

Sys\_level

This indicates the level of knowledge that the considered user has of the system in which the task must be performed.

#### **Output messages**

None

#### **Actions on the IS**

For each question, the IS contains the usual answers associated with them. After determining which usual answers can be applied to certain question, they must be recorded.

### **Treatment rules**

According to the levels of knowledge of the task and the system (revealed in the user profile) and according to the different levels of visibility, noise and treatment allocation (revealed in the context of work), some usual answers must be proposed when the analyst wants to answer the questions. So, these answers must be determined. For instance, if users have a good knowledge of the system, the usual answer « By knowledge of the system » will be proposed to the analyst. On the other side, if users have no knowledge of the system, it won't.

## **3.4. Goal/subgoals tree creation**

### **Objectives**

This functionality permits the creation of the decomposition tree of the task (the hierarchy of goals). For the atomic subgoals, the analyst must enter the sequence of actions composing them (in common language or in UAN). A list of common UAN actions will be provided to accelerated the capture of the actions. We have to let the opportunity to the analyst to modify some information attached to a goal or to change the structure of the tree ; so, not only operations of goal creation are to be implemented but also operations of updating, deleting.

### **Input messages**

Tree =

$\{\text{non\_atomic\_goal}\} \wedge \{\text{atomic\_goal}\}$

The tree is composed of a set of non atomic subgoals and atomic subgoal.

Non\_atomic\_goal =

$\text{Sequence\_num} \wedge \text{Name} \wedge \text{Description}$

Atomic\_goal =

$\text{sequence\_num} \wedge \text{name} \wedge \text{description} \wedge \text{actions\_seq}$

Actions\_seq =

$\{\text{action}\}$

action =

num  $\wedge$  value  $\wedge$  system\_feedback  $\wedge$  system\_status

### **Output messages**

None

### **Actions on the IS**

Creation of goals with the necessary information either for the atomic or the non-atomic, modification of the information related to a subgoal and deletion of a subgoal.

### **Treatment rules**

If a subgoal is atomic, the analyst must be told to enter the sequence of actions. If it is not the case, the number of subgoal-children is to be entered to know how many subgoal descriptions, the analyst must enter.

## **3.5. Method Performing, results synthesising and problem management**

### **Objectives**

For each subgoal present in the IS, the subgoal-related questions are to be asked to the analyst, recorded and the answers have to be compared with the knowledge base to see if there may be a problem and how serious is this problem. The same must be done with the action-related questions.

To help and answer the questions, usual answers must be proposed.

### **Input messages**

Answer =

description  $\wedge$  value

For each answer, the affirmative or negative aspect must be given with a longer explanation of the answer.

### **Output messages**

Problem\_seq =

[problem]\*

The result of the performing of the method will be a list of all the problem recorded which can be achieved with the assistance of the knowledge base.

#### **Actions on the IS**

Recording, deleting of answers.

### **3.6. UAN Library Creation**

#### **Objectives**

Users are allowed to specify their own Uan actions libraries that are used to determine the sequence of actions for each atomic subgoal.

#### **Input messages**

Library\_name

This is the name to attribute to the library to create.

Uan\_actions =

{uan\_action}

The library contains a list of Uan action, each included in a string.

Uan\_file\_name

This is the name of the file where the Uan library will be saved

Path\_to\_file

This is the complete path where the file will be saved.

#### **Output message**

Err\_path

This message is displayed on screen if the path specified is invalid.

Err\_filename

This message indicates to users that the name they want to use is invalid (contains reserved characters).

#### **Actions on the IS**

Recording of a Uan library.

### 3.7. UAN Library Saving / Loading

#### Objectives

To save or load the content of a UAN file.

#### Input messages

Uan\_file\_name

This is the name of the file where the Uan library is saved or where the Uan library has to be saved.

Path\_to\_file

This is the complete path where the file is saved or has to be saved.

#### Output message

Err\_saving

If the location and/or the name of the file are invalid, this message informs users that the saving was not performed.

∨

Err\_loading

If the location and/or the name of the file are invalid, this message informs users that the loading was not performed.

∨

Saving\_OK

This message informs users that the saving of the Uan library was performed.

∨

Loading\_OK

This message informs users that the loading of the Uan library was performed and that the library is now available for the specification of the atomic subgoal (i.e. the sequence of actions).

### 3.8. CW Saving/Loading

## **Objectives**

To save or load the content of a CW file.

## **Input messages**

CW\_file\_name

This is the name of the file where the CW file is saved or where the CW file has to be saved.

Path\_to\_file

This is the complete path where the file is saved or has to be saved.

## **Output message**

Err\_saving

If the location and/or the name of the file are invalid, this message informs users that the saving was not performed.

∨

Err\_loading

If the location and/or the name of the file are invalid, this message informs users that the loading was not performed.

∨

Saving\_OK

This message informs users that the saving of the CW file was performed.

∨

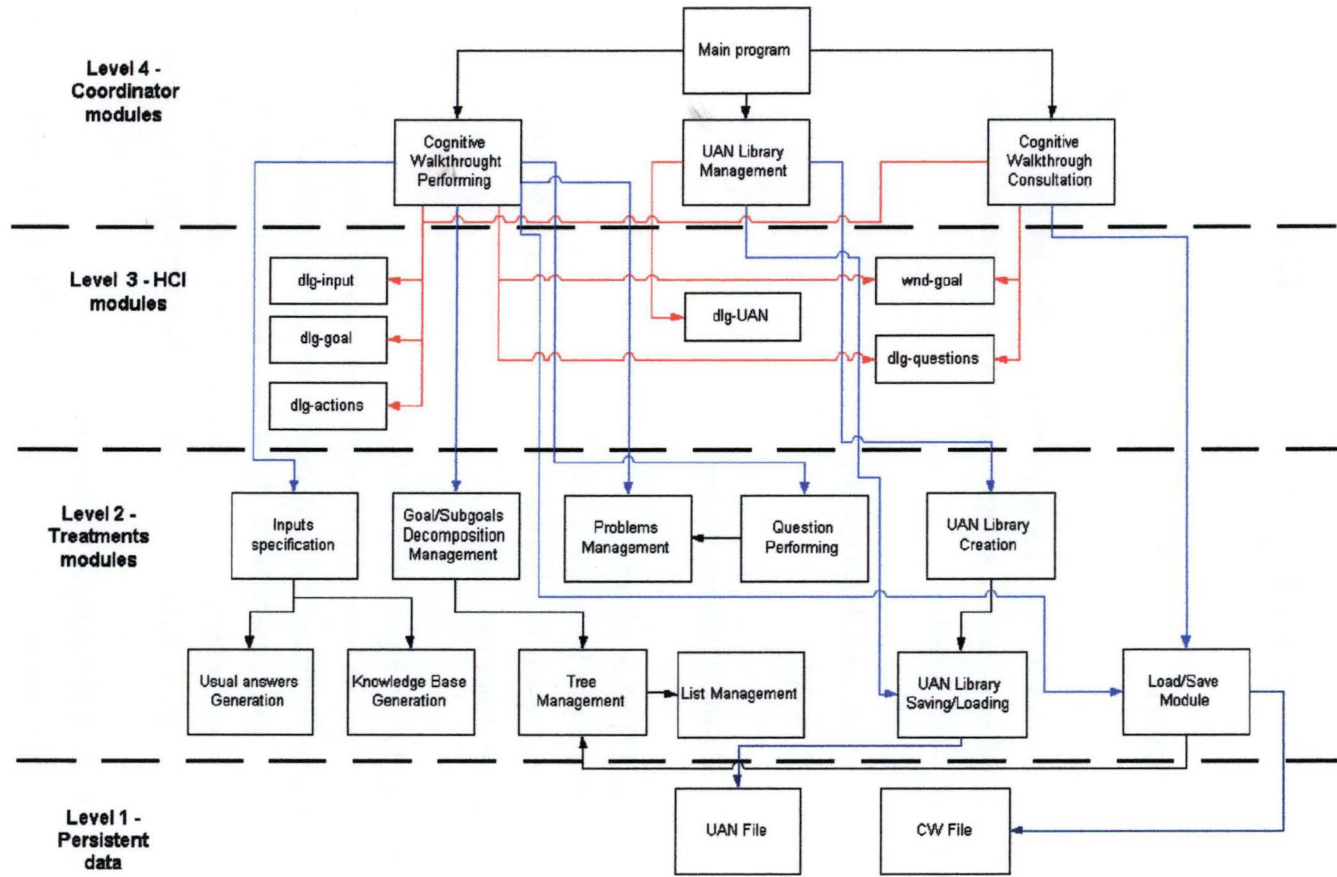
Loading\_OK

This message informs users that the loading of the CW file was performed.

## **4. Global architecture**



Figure 15 - Global architecture of the CW editor



## 5. Specification of the treatment modules

### List Management

#### Module description

This module provides a set of tools permitting the definition of the abstract type « index-sorted list » : creation, suppression of a list ; insertion, suppression and modification of an element.

#### Functional interface and description

Create\_list : \_  $\rightarrow$  tlist[T]

%Pre : \_

%Post : an empty list is created

Empty\_list : tlist[T]  $\rightarrow$  boolean

%Pré : the input includes a list

%Post : the result is a boolean whose value is true if the list is empty and false in the other case

Insert\_list : tlist[T] \* Integer \* T  $\rightarrow$  tlist[T]

%Pre : a list , an integer corresponding to the index and an element of the type T, representing the information field, are received.

%Post : If no element with the same index was present in the list, a new element was created, with its Info field initialized with the value of the third parameter and the so-got list is still sorted by increasing order.

Del\_elem\_list : tlist[T] \* Integer  $\rightarrow$  tlist[T]

%Pre : a list , an integer corresponding to the index are received.

%Post : If the index passed as parameter id present in the set of indexes of the list, the the element with this index was suppressed.

Delete\_list : tlist[T]  $\rightarrow$  tlist[T]

%Pre : A list is received

%Post : The list is empty.

Read\_list : tlist[T] \* Integer  $\rightarrow$  T

%Pre : a list and an integer representing an index are received.

%Post : if an element of the list has the same index as the one given as parameter, then the Info field of this element is sent as result. If not a NIL pointer is sent.

update\_list : tlist[T] \* Integer \* T → tlist[T] \* boolean

%Pre : a list , an integer corresponding to the index and an element of the type T, representing the information field, are received.

%Post : The value of the boolean is true if the update has been realized (if the index belongs to the indexes set of the list) and the Info field of the element with this index has been replaced by the one given as parameter. In the other case, the boolean value is false and the list remains unchanged.

## Types interface

Telem = UNION

[Index : Integer ;  
Info : T ;  
Next : ^Telem ;]

Tlist[T] = ^Telem

T is a generic type

## Tree management

### Module description

This module aims to provide all the necessary functions to permit the management of an abstract type « Tree ».

### Functional interface and description

Create\_tree : \_ → Tree[T]

%Pre : \_

%Post : an empty tree of T-type elements has been created.

Empty\_tree : Tree[T] → Boolean

%Pre : The input parameter is a tree of T-type elements.

%Post : the result is a boolean whose value is true if the tree is empty and false otherwise.

**Insert\_tree** : Tree[T] \* T \* Integer → Tree[T]

%Pre : an index, a tree of T-type elements and a T-type element are given.

%Post : the T-type element became a new leaf of the tree. This leaf received the index given as parameter ; this index is unique among the leafs of the tree.

**Delete\_elem\_tree** : Tree [T] \* Integer → Tree[T]

%Pre : an index, a tree of T-type elements are given.

%Post : if the index given is one of the tree leafs, then this leaf has been removed. Otherwise the tree remains unchanged.

**Delete\_tree** : Tree[T] → Tree[T]

%Pre : a tree of T-type element is given.

%Post : the tree is empty, all the leafs and the root have been deleted.

**Read\_tree** : Tree[T] \* Integer → T

%Pre : a tree of T-type element and an integer representing an index are received.

%Post : if the index was one of the leafs of the tree, then the T-type element corresponding to this index is returned. In the other case, only the NIL pointer is sent back.

**Update\_tree** : Tree[T] \* T \* Integer → Tree[T]

%Pre : an index, a tree of T-type elements and a T-type element are given.

%Post : if a leaf corresponding to the index provided as parameter exists, then its information field has been replaced by the T-type element. Otherwise, nothing changes.

### **Types interface**

Tsubroot =

UNION[Info : T ;  
Subtree : ^Tree ;]

Tleaf = Tlist[Tsubroot] ;

Tree = ^Tleaf ;

### **Uses**

List Management.

## Knowledge Base Generation

### Module description

The goal of this module is to construct a matrix crossing the usability and utility criteria ratings with the different Cognitive Walkthrough questions ; this permits to know when an answer is given to a question, if this answer reflects a potential problem or not.

Note : In the final version of our thesis, we have simplified the matrix. This one doesn't differentiate the different questions in the CW. The one we have specified takes in account the different questions according to the utility and usability criteria.

### Functional interface and description

Knowledgebase\_gen : Tlist[Crit\_Rating] → Tlist[CritQuest]

%Pre : the given list contains for each utility and usability criteria the rating associated to it.

%Post : The result is a list where is recorded the possibility, for each criteria and for each possible answer (Yes, No, Perhaps) of each question (CW1 to CW7), of a problem to occur.

### Types interface

Crit\_Rating =

UNION [criteria : ENUM[ LEARNING TIME, RAPIDITY OF EXECUTION, ERROR RATE, PERSISTENCY, SATISFACTION OF THE USER, COVERAGE] ;

Rating : ENUM[LOW,AVERAGE,HIGH];]

CritQuest =

UNION [criteria : ENUM[ LEARNING TIME, RAPIDITY OF EXECUTION, ERROR RATE, PERSISTENCY, SATISFACTION OF THE USER, COVERAGE] ;

VsQuestion : Tlist[Seriousness \* Seriousness \* Seriousness] ;]

Seriousness = ENUM [FAILURE, IMPORTANT, LIGHT]

### Uses

List Management

## Usual answers generation

### Module description

This module aims to generate the different usual answers required for the Cognitive Walkthrough questions according to the user profile and the context of work.

### Functional interface and description

Usualans\_gen : Task\_exp \* Sys\_exp \* Visibility \* Noise \* Treatment\_alloc → Tlist[questions\_ans]

%Pre : the five given parameters are to be initialized and represent the level of task, the system knowledge, the level of visibility, the level of noise and the treatment allocation.

%Post : The result lists the seven Cognitive Walkthrough questions. For each, are given the most likely answers by order of probability.

### Types interface

Question\_ans = Tlist[String] ;  
Task\_exp = Enum [LOW, AVERAGE, HIGH] ;  
Sys\_exp = Enum [LOW, AVERAGE, HIGH] ;  
Visibility = Enum [POOR, AVERAGE, GOOD] ;  
Noise = Enum [QUIET, AVERAGE, NOISY] ;  
Treatment\_alloc = [SINGLE, MULTI] ;

### Uses

List Management.

Inputs specification

## UAN Library Saving/Loading

### Module description

The goal of this module is to load the default and user-defined files including the UAN actions and to save to file the UAN library defined by the user.

### Functional interface and description

UAN\_load : String → Tlist[String]

%Pre : the input String represents a path to a UAN file.

%Post : the list contains all the UAN actions included in the opened file.

UAN\_Save : Tlist[String] \* String → \_

%Pre : the list contains the UAN actions of a user defined library and the string is the path-name-of-the-file to use.

%Post : the content of the list has been saved in a UAN file with the path-name-of-the-file given in input.

#### Uses

List Management.

## UAN Library Creation

### Module description

This module must permit the user to define his own UAN library and to update former user-defined library.

### Functional interface and description

UAN\_create : Tlist[String] \* String \* String → \_

%Pre : the list contains the UAN actions of a user defined library, the first string is the name of the file and the second one is the path-name-of-the-file to save.

%Post : the content of the list has been saved in a UAN file with the path-name-of-the-file given in input.

#### Uses

List Management.

UAN Library Saving / Loading.

## CW Saving/ Loading

### Module description

The goal of this module is to load a Cognitive Walkthrough from a file and to save it to a file.

### Functional interface and description

**CW\_load** : String → Tree[SUBGOAL] \* Tlist[Crit\_Rating] \* User\_profile \* Context\_of\_work \* Tlist[questions\_ans] \* Interface \* Tlist[Problem]

%Pre : the input String represents a path to a CW file.

%Post : the result contains the decomposition in goal/subgoals of the task selected in this CW, the balance of the utility and usability criteria, the user profile, the context of work, the list of the usual answers and the list of problems revealed.

**CW\_Save** : Tree[SUBGOAL] \* Tlist[Crit\_Rating] \* User\_profile \* Context\_of\_work \* Tlist[questions\_ans] \* Interface \* Tlist[Problem] \* String → \_

%Pre : all the elements presents in a CW and the string is the path-name-of-the-file towards the file to load.

%Post : the content of the elements of a CW has been saved in a CW file with the path-name-of-the-file given in input.

### Types interface

SUBGOAL = Integer \* String \* String \* Boolean \* Tlist [ANSWERS SUBGOAL] \* Tlist [ACTIONS] ;

ANSWERS SUBGOAL = String \* Answer ;

Answer = Enum [YES, NO, PERHAPS] ;

ACTIONS = Integer \* String \* String \* String \* Tlist [ ANSWERS ACTION];

ANSWERS ACTION = String \* ANSWER ;

User\_profile = String \* Task\_exp \* String \* Sys\_exp \* String ;

Context\_of\_Work = String \* Visibility \* Noise \* Treatment\_alloc ;

Interface = Integer \* String \* String ;

Problem = Integer \* String \* String \* String \* String \* Seriousness \* String

### Uses

List Management.



## Goal/subgoals tree creation

### Module description

The goal of this module is to make the decomposition in goal/subgoals/actions of a CW and to display it on the screen.

### Functional interface and description

Add\_Nonatomic : String \* String \* String \* Tree[SUBGOAL] → Tree[SUBGOAL]

%Pre : the input string(s) represents the level of hierarchy in the decomposition in goal/subgoals and the input String represents the goal, subgoal or action to put into the tree of decomposition in goal/subgoals.

%Post : the decomposition of the tree in goal/subgoals/actions until now.

Add\_Atomic : String \* String \* String \* Tree[SUBGOAL] \* ACTIONS → Tree[SUBGOAL]

%Pre : the input string(s) represents the level of hierarchy in the decomposition in goal/subgoals and the input String represents the goal, subgoal or action to put into the tree of decomposition in goal/subgoals. The last parameter is a list of actions composing the atomic subgoal to add to the tree.

%Post : the decomposition of the tree in goal/subgoals/actions until now.

Display\_decomposition : Tree[SUBGOAL] → \_

%Pre : the decomposition of the tree in goal/subgoals/actions until now.

%Post : The decomposition in displayed on the screen.

### Uses

Tree Management

## Method Performing, results synthethising and problem management

### Module description

For each subgoal present in the IS, the subgoal-related questions are to be asked to the analyst, recorded and the answers have to be compared with the knowledge base to see if there may be a problem and how serious is this problem. The same must be done with the action-related question.

To help and answer the questions, usual answers must be proposed.

### Functional interface and description

Add\_subgoal\_ans : Answer \* String \* Answer \* String \* Tree[SUBGOAL] → Tree[SUBGOAL]

%Pre : the two subgoal-related answers and their description are given with the decomposition tree.

%Post : the subgoal-related answers have been added to the corresponding subgoal in the tree.

Add\_action\_ans : Answer \* String \* Answer \* String \* Answer \* String \* Answer \* String \* Answer \* String \* Tree[SUBGOAL] → Tree[SUBGOAL]

%Pre : the five action-related answers and their description are given with the decomposition tree.

%Post : the action-related answers have been added to the corresponding atomic subgoal in the tree.

Test\_problem : Tlist[CritQuest] \* Tlist[ANSWERS SUBGOAL] \* Tlist[ANSWERS ACTION] → Tlist[Problem]

%Pre : the knowledge base and the answers are given.

%Post : If a problem is likely to occur (according to the answer and the base knowledge), it has been added to the problem list.

### Uses

Tree Management.