

NOTE TO USERS

This reproduction is the best copy available.

UMI[®]

A Usability Study of
Two Knowledge Acquisition Tools

Mohammad Lokman Hossain

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

April 2005

© Mohammad L. Hossain, 2005



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*

ISBN: 0-494-04446-2

Our file *Notre référence*

ISBN: 0-494-04446-2

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

ABSTRACT

A Usability Study of Two Knowledge Acquisition Tools

Mohammad Lokman Hossain

In today's world, knowledge-based systems are widely used for both large and small tasks. Acquiring knowledge from domain experts is one of the major bottlenecks in building such systems, and hence, Knowledge Acquisition (KA) tools are given importance. The goal of these tools was to assist the knowledge engineer in eliciting knowledge from domain experts. Until recently, the usage of these tools was limited to people with significant computer expertise. User interface difficulties form one source that prevents people at large from receiving the expected benefits of KA tools. This has prompted the investigation of the usability of two popular knowledge acquisition tools in this thesis.

This thesis focuses on designing and conducting a usability test with two popular KA tools: Protégé2000 and PCPACK4. The purpose of this study is to assess the usability of these two KA tools against a set of criteria, and to provide recommendations for further improvement of these tools or to help in the development of new KA tools. To conduct this usability test, this thesis adopted and modified Hix's methodology for the evaluation of knowledge-acquisition tools. Testing with a small set of end-users revealed several usability issues. Based on these observations recommendations are made in this thesis for further enhancement of usage of KA tools.

Dedicated to my parents, Assad and Aziba.

ACKNOWLEDGMENTS

My deepest gratitude goes to my supervisors, Dr. T. Radhakrishnan and Dr. Ahmed Seffah for their invaluable guidance and encouragement. I consider Dr. Radhakrishnan much more than a supervisor. He is a father figure who cares very deeply about the welfare of his students. The financial support for this work has been provided by the Canadian Institute for Telecommunication Research special project on Highly Qualified Personnel Training (HQP) through a research grant to Dr. T. Radhakrishnan. The support from CITR is greatly acknowledged.

I would like to thank my colleagues in Human Computer Interaction Lab for their cooperation during this time. I would like to thank the test participants for their cooperation during the testing.

Thanks to my family members and friends for their help and support. I like to give a special thanks to my wife Yasmin who was always there whenever times became difficult. I deeply appreciate her care and support during my studies at Concordia.

TABLE OF CONTENTS

List of Figures	ix
List of Tables.....	X
1. Introduction.....	1
1.1 Problem Statement.....	1
1.2 Related Work.....	2
1.3 Investigating The Usability of KA Tools: A new Approach...	3
1.4 Objectives and Scope of the Thesis.....	4
1.5 Thesis Organization.....	6
2. An Analysis of Two Knowledge Acquisition Tools: User Interface Prespective.....	7
2.1 PCPACK4.....	7
2.1.1. Important Features and Functionalities.....	8
2.1.2. PCPACK4 Usage: An illustrated Scenario.....	9
2.2. Protégé 2000.....	16
2.2.1 Important Features and Functionalities.....	17
2.2.2. Protégé Usage: An illustrated Scenario	18
2.3. Summary.....	24
3. Techniques and Methods of Usability: An Overview.....	25
3.1. What is Usability?.....	25
3.2. Usability Measures.....	26
3.3. Usability Evaluation.....	29
3.3.1 Usability Inspection.....	30
3.3.2 Usability Inquiry.....	32
3.3.3 Empirical Testing.....	34
3.4. Selection of Usability Technique and Methods.	36

4. Usability Testing of Knowledge Acquisition Tools:	
A New Approach.....	39
4.1.Hix’s Checklist-based Methodology.....	39
4.2.Our Objectives.....	41
4.3. Modified approach to evaluate Protégé and PCPACK KA tools....	42
4.3.1. Selection of the test tasks and participants.....	42
4.3.1.1. Selection of tasks.....	43
4.3.1.2. Selection of participants.....	48
4.3.2. Creation of the tables.....	49
4.3.3. Learning the tools.....	53
4.3.4. Performing the tasks.....	53
4.3.5. Review of the sessions and completing the table entries	54
4.3.6. Compute the metrics.....	56
4.4. Summary.	59
5. Analysis and Interpretation of Test Results.....	60
5.1. Results.....	60
5.1.1. Participants’ Expertise.....	60
5.1.2. Test Results.....	61
5.1.3. Satisfaction Questionnaire.....	66
5.1.4. Problems.....	68
5.2. Analysis of Usability Problems.....	71
5.2.1. Classification of Usability Problems.....	71
5.2.2. Interpretation of Usability Issues	74
5.2.3. Usability Principles.....	84
5.3. Lessons Learned.	86
5.4. Recommendations	87
5.5. Discussions of the Results.....	91

6. Conclusion	94
6.1. Contributions.....	94
6.2. Future Works.....	95
References	96
Appendix	103

LIST OF FIGURES

Figure 2.1: Tool Launcher in PCPAK4 tool-set.....	10
Figure 2.2: Initial screen for creating PCPACK knowledge base.....	11
Figure 2.3: Protocol tool for analyzing the text.....	11
Figure 2. 4: A rearranged PCPAK4 ladder for students.....	12
Figure 2.5: Matrix attributes in the Y-axis, and concepts in X-axis.....	13
Figure 2. 6: Knowledgebase Matrix after modification.....	14
Figure 2.7: PCPACK4 Code for creating the Annotation template.....	15
Figure 2.8. Annotation Template for Graduate Student Node.....	15
Figure 2.9: Annotation corresponding to Masters Node.....	15
Figure 2.10: Initial screen for creating a new project in Protégé 2000.....	19
Figure 2.11: A standard text project in Protégé-2000.....	19
Figure 2.12: Protégé class tab for creating and modifying class.....	20
Figure 2.13: Slot tabbed window for creating and modifying slots.....	21
Figure 2.14: Form-tabbed window for customization.....	21
Figure 2.15: Instance tab for browsing and inserting Instances.....	22
Figure 2.16: Query generating tab.....	23
Figure 4.1: Hierarchical task analysis to derive the test tasks.....	46
Figure 5.1: Participants computer expertise.....	61
Figure 5.2: Usability metrics for Protégé and PCPACK.....	66
Figure 5.3: Protégé metaphors in save dialogue.....	75
Figure 5.4: Protégé user interface.....	76
Figure 5.5: PCPACK ladder toolbar icons.....	82

LIST OF TABLES

Table 3.1: Satisfaction questionnaire for Protégé and PCAPCK KA tools.....	28
Table 3.2. Usability Evaluation Methods Overview	29
Table 3.3: Definitions of heuristics.....	32
Table 4.1: Descriptions of technical terms used in figure 4.1.	47
Table 4.2: Task List for Protégé 2000 and PCPACK4.	48
Table 4.3: Evaluation table for Protégé KA tool.....	51
Table 4.4: Evaluation table for PCPACK4 KA tool.....	52
Table 4.5: Evaluation table of a particular user of Protégé KA tool	55
Table 4.6: Combined evaluation table (Partial) of Protégé KA tool.....	57
Table 5.1: Combined table for Protégé KA tool.....	63
Table 5.2: Combined table for PCPACK KA tool.....	64
Table 5.3: User's comment on difficult and confusing aspect.....	67
Table 5.4: Worst aspects according to user comment.....	67
Table 5.5: Protégé and PCPACK Satisfaction scores.....	68
Table 5.6: Usability issues for Protégé and PCPACK.....	74

1. Introduction

1.1. Problem Statement

Knowledge Acquisition (KA) tools are used to provide support in eliciting, and transferring expertise from a knowledge source (i.e. human expert, documents) to a computer program (Hayes-Roth, 1983). Such tools are gaining wide popularity in the industry today. Application of these tools range from creating and maintaining expert systems to knowledge based Internet sites.

The idea of using knowledge acquisition tools was started in late 1970. In the beginning, the aim of knowledge acquisition tool was to assist knowledge engineer by providing direct interaction with the domain expert, and some of the early tools (i.e. TEIRESIAS [Davis, 1979], MORE [Kahn, 1984]) were limited in scope when used by a knowledge engineer for data inserting, error checking, and data editing. During Mid 1990, a number of KA tools in the market place were claiming the ability to capture knowledge by interacting with the domain experts very successfully, but researchers noticed that some tools are still restricted by implementation environment rather than end-users [Puerta. A.R., and Erickson. H., 1994]. With the advancement of software engineering tools and technologies, more advanced KA tools (Protégé 2000, PCPACK4, WEBGRID III) have been developed, which are easy to use for a knowledge engineer or even by a domain expert who has adequate computer experience. It would be ideal for a domain expert to play the role of a knowledge engineer rather than an intermediary playing that role. So, in order to overcome the communication barrier, these tools should be adequately user-friendly.

The impediments for having a user-friendly KA tool are numerous, such as (1) users with less computer skill take more time to learn the KA tools; (2) terminologies are knowledge engineer based not domain based; and (3) most of the recent tools' user interfaces are with too many complex functionalities and features. Understanding all the functions, and features, and applying them correctly takes time and effort, and also, the users have to remember many details, which loads their short-term memory. In this perspective, Usability, a competitive necessity for the success of software [Butler, 1996], needs to be achieved for having successful knowledge acquisition. Either to improve the existing tools, or to create a new usable tool, human computer interaction problems that exist in today's KA tool needs to be discovered through proper usability study.

In this thesis, two KA tools (Protégé and PCPACK) will be evaluated from user interface perspective. The evaluation results can possibly lead to a set of benchmarks for further study of KA tools and recommendations for the improvement of these tools or to develop more usable KA tools.

1.2. Related Work

The literature related to the usability studies of knowledge acquisition tools is not substantial. In the following, we are providing some of the related works that have been done.

- [Kim et. al, 2000] conducted a series of experiment with a range of users to evaluate an intelligent interface for acquiring problem solving knowledge to describe how to accomplish a task. They reported that EMeD (A Method

Acquisition Interface for the EXPECT Architecture) saves users an average of 32% of the time it takes to add new knowledge.

- TURVY study (Maulsby et al, 1993) was conducted by David Maulsby and Allen Cypher at the advanced Technology group at Apple Inc. It tests an approach to programming by demonstration that learns as a user performs simple tasks.
- [Noy N.F. et al. 2000] conducted an empirical study to test the Protégé 2000 KA tools performance by domain experts. As developer of Protégé, they tried to test certain selected criteria of usability, such as ability to find errors in knowledge base, rate of knowledge acquisition, errors occurring during the task execution etc.

In this thesis, we concentrate on the methodology and design of usability experiments to investigate the usability measures and problems of two KA tools (Protégé 2000, and PCPACK4). The results can be used as a guideline for further design of knowledge acquisition tool.

1.3. Investigating the Usability of KA Tools: A New Approach

Knowledge Acquisition tools belong to a class of interactive systems that provides an environment for developing knowledge bases. Despite recent proliferation of such tools, it is believed that there are no procedures for systematically evaluating the usability of KA tools. The closest tool evaluation research was the “Checklist-based methodology (Hix, et. al. 1991) for evaluating and comparing human computer interface development tools.” This research does not specialize to KA tools but is general for all interactive

systems. We follow this methodology and make certain minor modifications and apply them to the study of two selected KA tools.

This approach is based on a number of representative participants, tasks, and evaluation criteria relevant to the KA tools (Protégé and PCPACK). Tasks have been selected based on task analysis. Representative users have been recruited by analyzing the user profile of the target audience. Before testing, the author as the evaluator had prepared an “evaluation table” using selected criteria as conditions, and simple task¹ as actions. Having prepared the necessary groundwork, test monitor scheduled the test sessions with ten test participants. Before test session, the participants were given a short tutorial so that they become uniformly acquainted with the tool environment. Test session was recorded by screen capturing software “Camtasia” with the consent (Appendix A3) of the participant. By reviewing the recorded session, the evaluation table entries were filled out by the evaluator. At the same time, evaluator also noted some other critical incidents from the test session. At the end of the session, a small post-test questionnaire (Appendix A5) was given to the participants for getting their assessment about the tool.

1.4. Objective and Scope of the Thesis

The purpose of this thesis is to assess the usability of two KA tools Protégé and PCPACK. Before defining the test objectives, it is important to understand the backgrounds of the users who will use these KA tools. Domain experts and knowledge engineers use knowledge acquisition tools to create and maintain knowledge-based system. The primary end-users of these tools are domain experts. So, the success of these

¹ Simple Task: the simplest unit of task, which has no control structure.

tools depends on the usefulness of these tools by the domain experts. Considering the above, we are aiming to test the following objectives with domain experts (end-users), which in turn will fulfill the broad objective of our evaluation.

1. To assess usability measures (Learnability, Understandability, and Satisfaction) of Protégé and PCPACK KA tools.
2. To find usability problems in the user interfaces of these tools to provide recommendations for the improvement of KA tools.

From several existing knowledge acquisition tools, the two popular KA tools Protégé and PCPACK were specifically chosen because they are perceptibly similar in structuring the knowledgebase; both use ontology as the base of the knowledgebase. Therefore, it would be expected to perform task in similar way rather than being different. As a result, common problems across these tools can be collected, consequently, which can be used to provide recommendations for further development of similar knowledge acquisition tool. In this study, many issues arose regarding internal and external characteristics of these tools. However, our research aimed only at answering the user interface related following questions:

- How to design an effective test to measure the usability of a knowledge acquisition tools?
- How to evaluate a knowledge acquisition tool?
- What are the major usability issues that should be addressed during the design of KA tools?

1.5. Thesis Organization

This chapter (chapter 1) started with the problem definition, and then the way to conduct the proposed study. Some related works to understand the background of the research were presented.

Chapter 2 provides an analysis of knowledge acquisition tools from the user interface perspective. Protégé 2000 and PCPACK4 are described to provide background information for understanding the results obtained from the study.

Chapter 3 discusses the concepts of usability testing, also discusses different candidate techniques and methods in the context of this study.

Chapter 4 presents the methodology to perform the usability study. It describes the major steps required to conduct the test, which includes recruiting participants, selecting tasks, criteria, developing evaluation table, performing the tasks, collecting the data and computing the metrics.

Chapter 5 provides the compiled and summarized results gathered from the tests, usability issues derived from the test result and recommendations made to overcome the problems that have been observed during the test.

Chapter 6 summarizes the contributions of the thesis particularly the usability issues of KA tools. It also discusses some future improvements of our studies.

2. An Analysis of Two Knowledge Acquisition Tools: User Interface Perspective

In this chapter, we analyze Protégé 2000 and PCPACK4, two popular knowledge acquisition tools. The analysis focuses on the features and functionalities of the user interface, and we use scenario-based examples for understanding the usability aspects.

2.1. PCPACK4

PCPACK4, a set of tools, was developed to create knowledge bases for any knowledge-based systems or knowledge-based Internet sites. Epistemics [Speel P.H. et al. 1999] developed PCPACK under a contract with the UK's Defense Research Agency. Knowledge engineers and domain experts use these tools for the following purposes:

1. Analyze knowledge from the text document.
2. Structure knowledge using different models.
3. Acquire and validate knowledge from various experts.
4. Publish and implement the knowledge base.
5. Re-use knowledge in different domains and subject areas.

KA tools make use of knowledge acquisition techniques and knowledge models for representing knowledge in the knowledge bases. PCPAK4 uses a number of techniques, such as Protocol analysis, Laddering, Diagram based, and Grid-based techniques for eliciting knowledge from the various knowledge sources, and it uses ontology templates for the initial structure of the knowledge base. If needed, the user of the tool can create a new ontology for creating the structure of a new knowledge base.

2.1.1. Important Features and Functionalities

1. Multiple windows can be opened and placed anywhere on the screen. If required, particular arrangement of windows can be selected.
2. There is an option to create new version of the existing knowledge base when it is loaded to different tools for adding and modifying the content of the knowledge base, and it keeps the original KB unchanged.
3. Different color markers are available to highlight the objects in the text transcripts.
4. Post-it notes can be attached to the marked objects. And, it is also possible to hide or view all post-it notes associated with the marked objects.
5. Different styles for each relationship link line can be used to enhance the presentation.
6. Matrix widget is used to represent the relationship between concepts and attributes. It is also possible to change the relationship by just a mouse-click in the matrix cell. The relationship is stored in the knowledge base.
7. Diagram templates can be used for specifying the styles of different types of nodes or links. Templates can be any network style diagrams where relationship exists among the nodes. In template, any node can be set to be decomposable to further lower level.
8. Realistic metaphors are available, such as marker toolbar with different color markers, and eraser, etc.

9. Help system is suitable for beginners as well as advanced users. Help system provides individual tutorial for each tool. An online help facility is available from the developer.
10. A window for browsing ontologies is available with almost all tools in PCPACK4, except annotation template, and annotation tools. Necessary objects can be dragged from ontology browsing window and dropped to tool-window
11. Flexibility to place the toolbar anywhere in the window
12. Publishing tool is available to create a standard website. All hyperlinks in annotation pages are maintained when published. The style, structure and content of the website can be customized.

2.1.2. PCPACK4 Usage: An illustrated Scenario

Problem: In this scenario, a knowledge-based system is considered that is intended for managing students' information. The scenario is presented in several steps. It involves retrieving students' information when needed, categorizes students based on different factors, such as grades, level of study, nationality, scholarship etc and analyzes students' data from different perspectives. For performing these activities the knowledge base needs to contain the following information:

- Information about the program and duration of study
- Information about students' grades, type of scholarship, nationality (local or international), current status of the student.
- Personal Information, such as name, address, phone number, date of birth etc.

Step 1 (Tool Launching): Tool launcher is an integrated environment from where other tools can be launched. At the same time, it can be used for creating knowledge base as well. For beginners, it provides a good starting point for knowledge acquisition activities. Tool launcher that is shown in figure 2.1 can be started from the navigation sequence “Programs→PCPACK4→Launcher”.

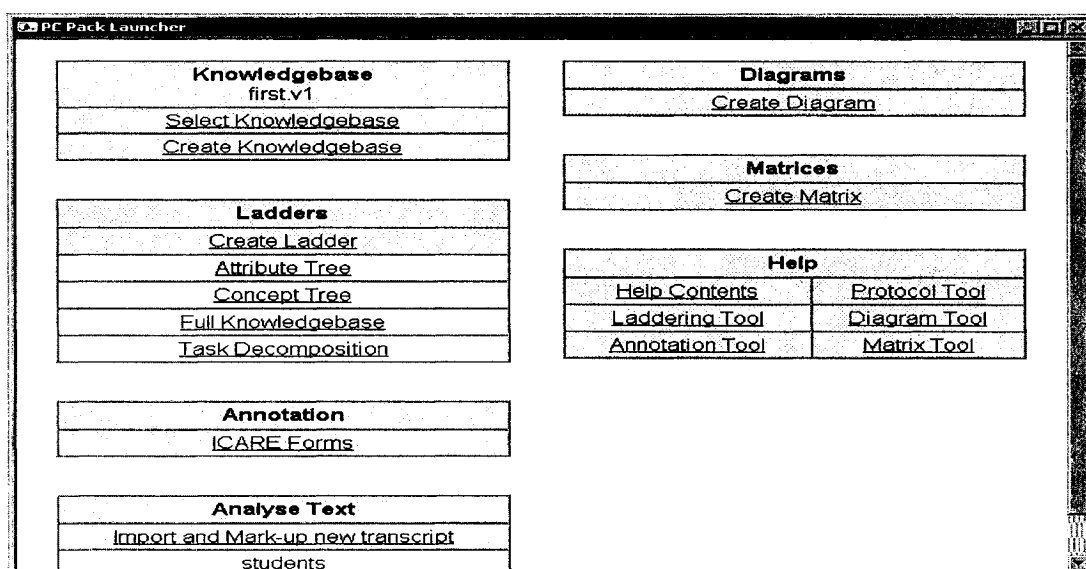


Figure 2.1: Tool Launcher in PCPAK4 tool-set.

Step 2 (Creating Knowledge Base): Knowledge base can be created either from the tool launcher (fig-2.1) or from any other tool in the tool-set except from the annotation tool. After choosing to create a new knowledge base, a window (figure 2.2) appears for taking the name, security password, and selecting ontology for the knowledge base. Empty ontology or any existing one can be chosen for the initial template of the knowledge base. After creating an empty knowledge base, any tool can be selected for knowledge acquisition activities. For example, for analyzing the existing knowledge (in the text file), Protocol tool from the popup start menu can be invoked.



Figure 2.2: Initial screen for creating PCPACK knowledge base.

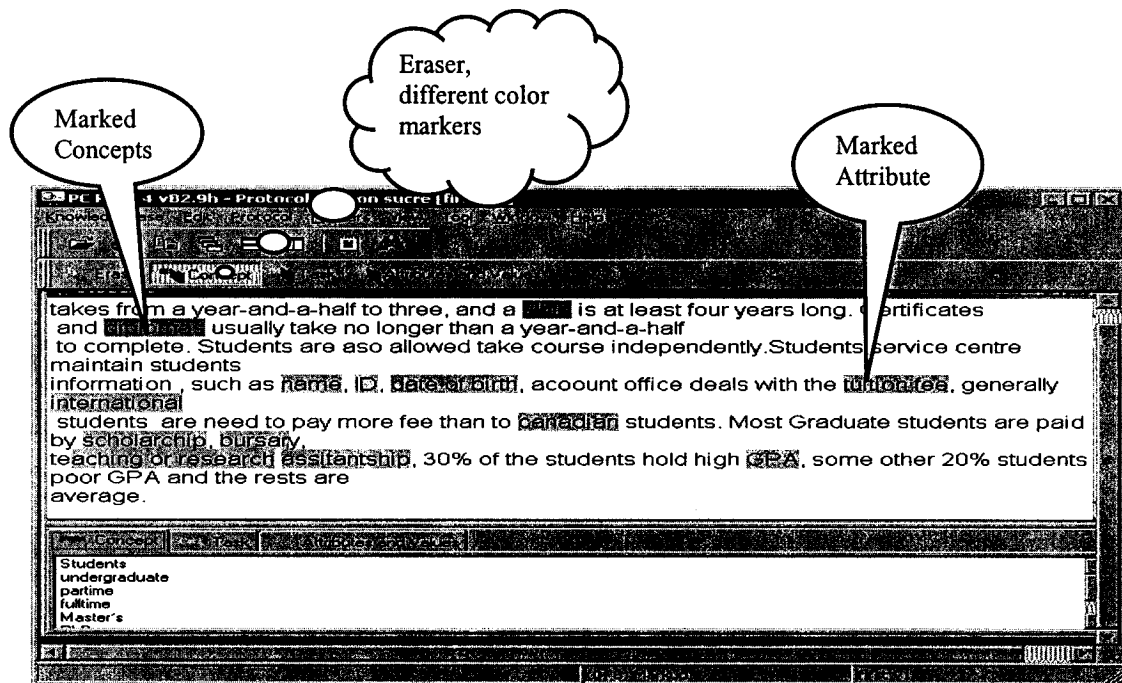


Figure 2.3: Protocol tool for analyzing the text.

Step 3 (Importing transcripts): The Protocol tool without any source file is just a blank window with menus and tool bars. For analyzing “students.txt”, which contains the transcripts of the knowledge base, needs to be imported to the integrated environment. It can be imported by selecting the “protocol --> import”. Figure 2.3 shows typical transcripts in the protocol window.

Step 4 (Analyzing the Text Transcripts): For analyzing the imported transcripts the protocol tool uses two default markers and one eraser to mark the objects in the window that is shown in Figure 2.3. If needed, different color markers can be added to mark different types of knowledge entities in the text. The new marker can be added from the marker pull down menu of top menu bar. The marked concepts and attributes are added to the knowledge base automatically.

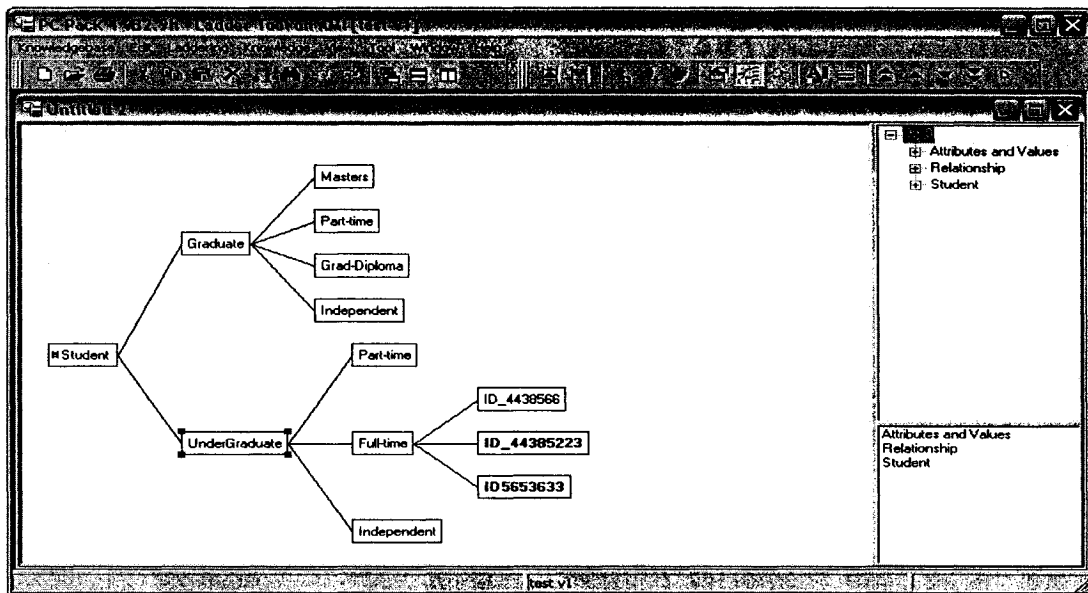


Figure 2. 4: A rearranged PCPAK4 ladder for students.

Step 5 (Creating the Ladder): Laddering tool is used to create and edit ladders, which are hierarchical (tree-like) representations of knowledge in the knowledgebase. Each ladder comprises nodes and links. Initially the ladder displays the knowledge items based on default ontology selected in the beginning, but the real relationship hierarchy among the knowledge objects might be different. For this reason, the ladder needs to be rearranged (figure 2.4). Making one node as the root of all similar types and other nodes as different level children of the root creates hierarchy in the ladder. These options are available from

toolbar or popup menu, which can be invoked by right-mouse click at the selected object in the ladder.

Step 6 (Creating the Matrix): Two types of matrix possible from the matrix tool: attribute matrix, and relationship matrix. Attribute matrix is a way of associating attributes and values to knowledge objects and Relationship matrix used to define the relationship between concepts in the domain. Selecting “create new matrix” option from the matrix pull-down menu of the top menu bar can create attribute or relationship matrix. Attribute matrix shown in figure 2.5 can be generated from default row, column, or it is possible to rearrange the row-columns to generate the matrix as needed. Similarly, row and column can be created from the concepts to create relationship matrix, which is not required in this example. Using this tool, new values can be added and relationship can be changed according to the requirement. Figure 2.6 shows one of the modified attribute matrix. The domain expert can validate the acquired knowledge by observing the matrix.

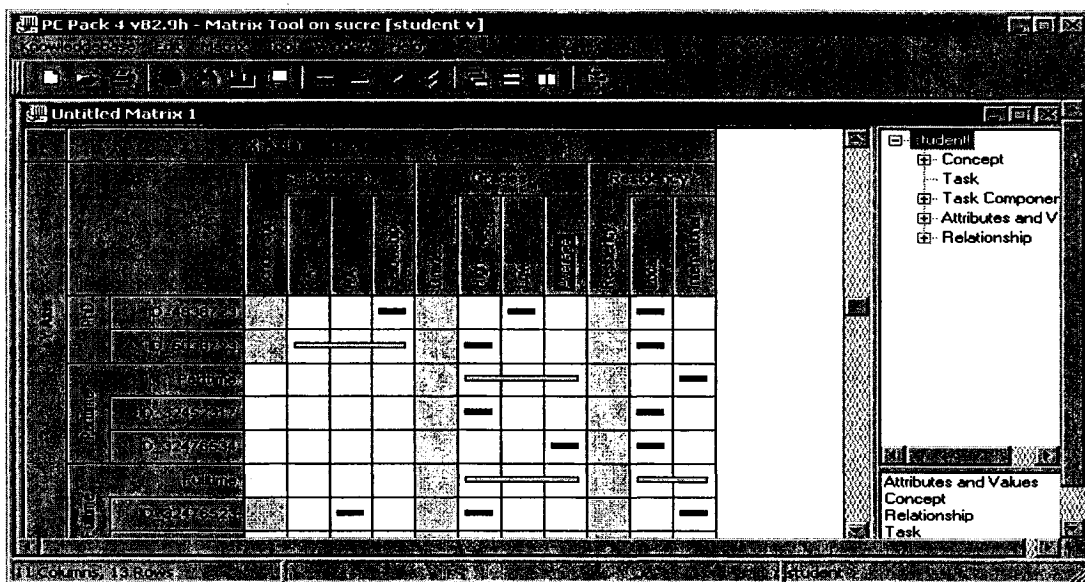


Figure 2.5: Matrix attributes in the Y-axis, and concepts in X-axis.

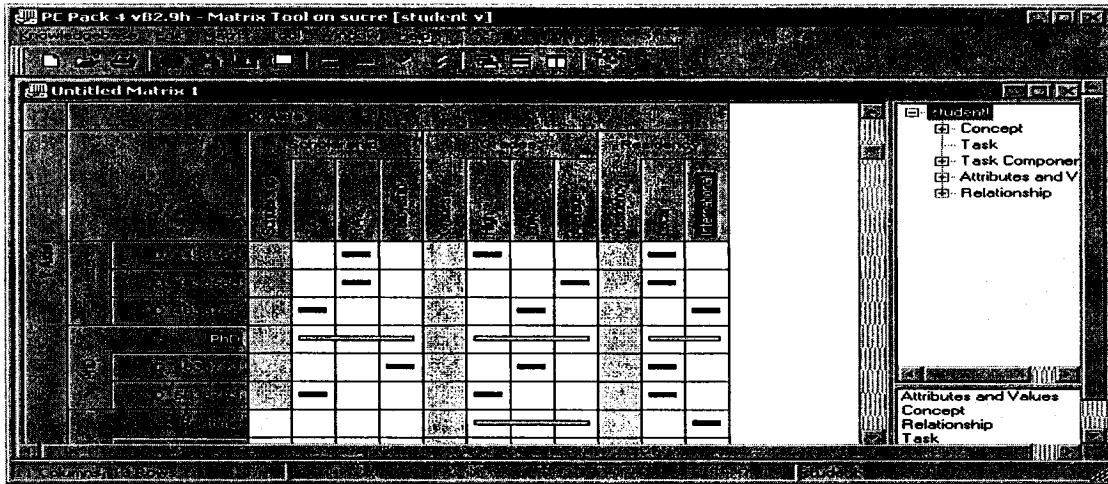


Figure 2. 6: Knowledgebase Matrix after modification

Step 7 (Creating The Annotation Template): Templates allow annotations for objects to have a consistent appearance, and it is the prerequisite for creating the annotation for an object. If a template is created for an object, all new annotations for descendants of the object will inherit it unless the descendant has a different template. Though object and relationship template are possible, current PCPACK supports only object annotation template. If a template is edited, annotation pages previously created with it will be unaffected. A typical template can contain relationship with the objects, the name of the objects, the attributes and relations associated with the particular objects. The node for which the template needs to be created is selected from the ladder. Then cascading popup menu provides option for selecting annotation template tool for creating corresponding template for the node. The template is inheritable to the node's subclasses. Figure 2.7 shows the annotation template for Graduate node in the ladder, and figure 2.8 displays the annotations for graduate object, and figure 2.9 shows annotation masters' node which was inherited from the parent node, other children node will also inherit the template.

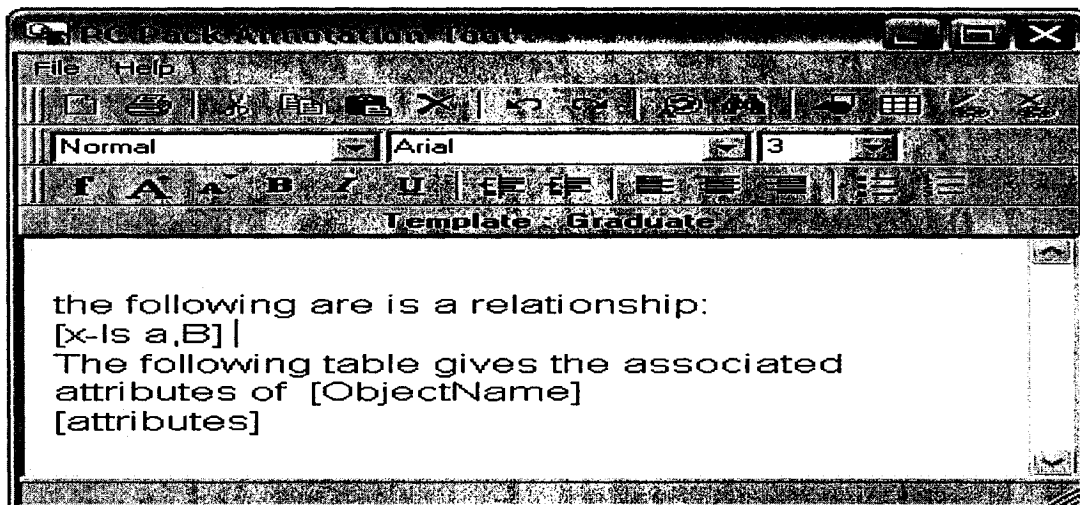


Figure 2.7: PCPACK4 Code for creating the annotation template.

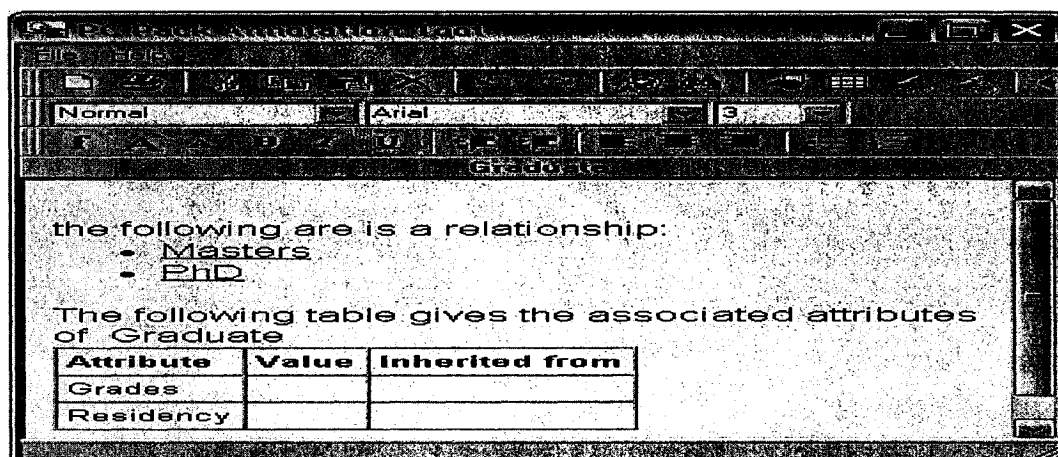


Figure 2.8. Annotation Template for Graduate Student Node.

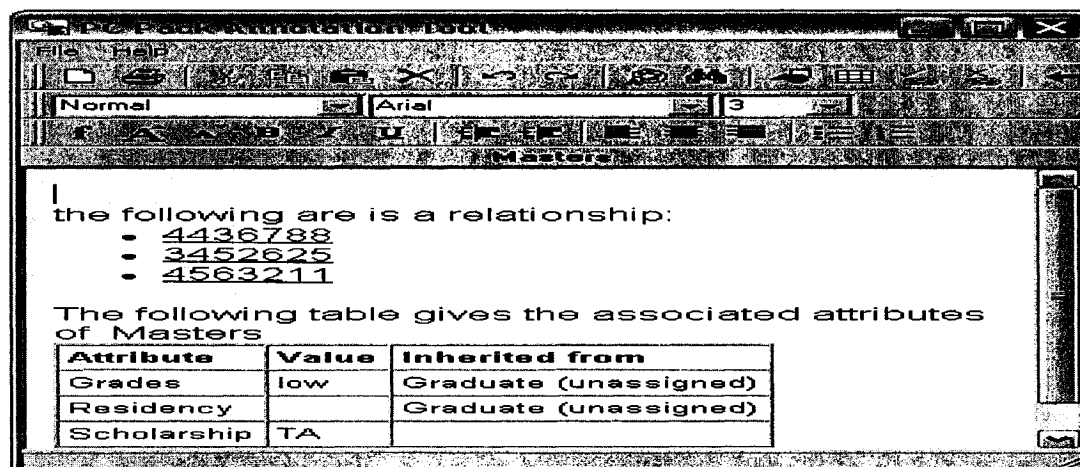


Figure 2.9: Annotation corresponding to Masters Node.

Step 9 (Publish the Knowledge Base): The knowledge base created by PCPACK4 tools normally published after the expert validation. Publishing is simple with the “ Publish Tool” menu. The knowledge is published in the default XML or in a selected format on the web.

Using the above scenario-based step-by-step method, I have created a sample knowledge base web site, and it is presented in Appendix B2. It is not mandatory to follow all the steps that have been described above. For example, if someone doesn't have any text transcript s/he doesn't need to use the protocol tool rather s/he can start from ladder tool to build the knowledge base.

2.2. Protégé-2000

Protégé-2000 is a tool that enables users to acquire knowledge and maintain the knowledge base [N. F. Noy, et al., 2000]. Application experts can use it to enter and browse knowledge base instances. Protégé uses domain ontology, which is a means to specify the domain conceptualization (Guarino, 1995), as the skeletal of the Knowledgebase [Cupit, J. et al., 1999]. In fact, Protégé is both an ontology development and knowledge acquisition tool, which is designed to provide assistance for knowledge engineers as well as domain experts in maintaining and editing knowledge bases. It is a component based, and platform independent tool-set, which supports Open Knowledge-Base Connectivity (OKBC) protocol [Chaudhri, V.K., 1998]. OKBC facilitates interoperability by providing an application-programming interface (API) that serves as common query and construction interface for frame-based systems. Developers can enhance the Protégé system by contributing new plug-ins to a library that is maintained

on the Internet. Users can freely download from the sites to augment the behavior of their own knowledge acquisition systems constructed using Protégé-2000.

2.2.1 Important Features and Functionalities

1. Widgets are used to display and input data of particular types in a task specific way. Protégé-2000, currently, supports two types of widgets, default widgets, which are generated with the default layout and advanced widgets, which are added to the Protégé-2000 project by loading special plug-ins.
2. Protégé uses Metaclass [Noy, et al., 2000], which is the template of ontology classes and slots. The usage of Metaclass makes the creation of classes and the ontology simpler.
3. Over lapping tabbed window feature (Shneiderman, 1992) is embedded in the user interface. It allows partial obscuring of each other like overlapping papers arranged on a desk. It places different tabs, such as Class, Slot, Form, Query and Instance to be represented in the over lapped way.
4. Popup menus are attached with all the selected items to change the relevant attributes of the items. It provides almost all-available options that can be used from toolbar and menu bar.
5. Customizable forms are available that are generated automatically corresponding to a class, where widgets correspond to slots in the class. These customized forms are used to insert instances by the end-users.

6. Different look and feel user-interface styles are supported. To facilitate major user groups, protégé provides three different look and feels, such as metal look and feel, windows look and feel, and java look and feel.
7. Diagrammatic approach is followed to display the ontology and its instances for visualizing the knowledge base contents and their relationship.
8. Wide varieties of data types (String, Boolean, Symbol, Instance, Class, Integer, and float) for slot definition are supported.
9. Allows importing ontology from compatible sources. For example, ontology constructed in the ontolingua can be imported to Protégé system.
10. Alternatives “back-ends” are provided for archival storage. In the early development, Protégé would store all ontologies and knowledge bases on flat files encoded in the CLIPS knowledge representation language, but protégé 2000 allows developers to add new “back-end” support for alternative storage. With the help of plug-in capability several storage formats (XML, RDF, and Relational database) have already been created, and it is also possible to add any new desired format by developing new plug-in.

2.2.2. Protégé 2000 Usage: An illustrated Scenario

Problem: In the previous section, we used a scenario for PCPAK4 and it is used also for illustrating Protégé 2000 in the same manner.

Step 1 (Creating the Project): The first task to develop a knowledge-based system using protégé involves creating a project. After choosing to create a new project

("Project→new" at fig-2.10), a window will appear on the screen for selecting the knowledgebase format. Based on the format selected, another window will appear to accept the project name and other information as shown in figure 2.11. In this example, standard text file has been selected; consequently, a window (figure 2.11) appears to take the project, class, and file name. After receiving the project name, same name will be automatically inserted with different extension, and an empty knowledge base will be created.

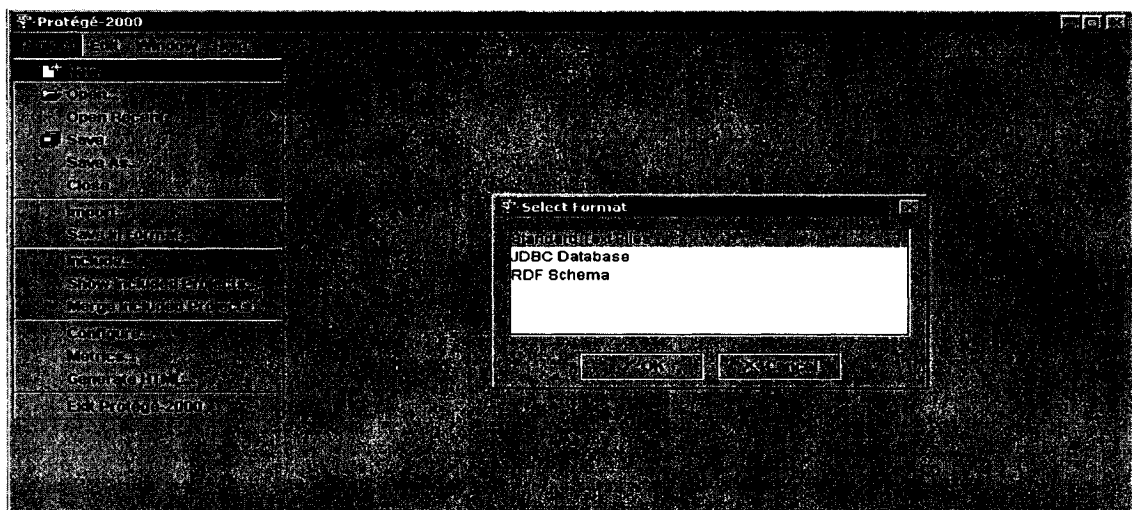


Figure 2.10: Initial screen for creating a new project in Protégé 2000.

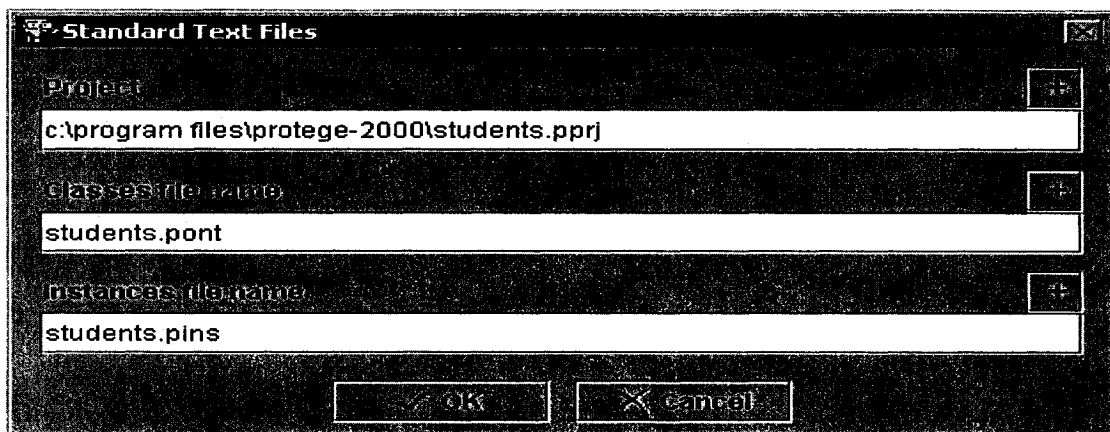


Figure 2.11: A standard text project in Protégé-2000.

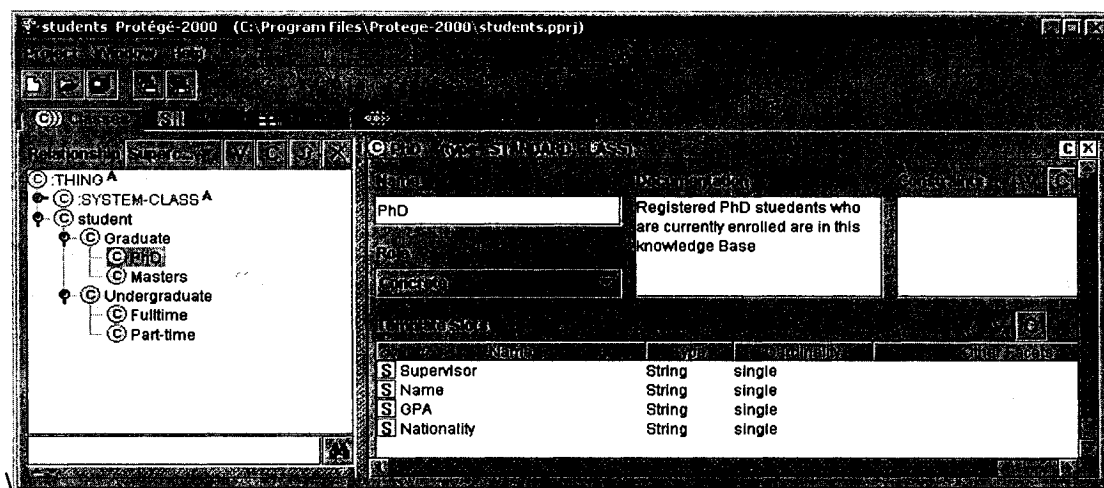


Figure 2.12: Protégé class tab for creating and modifying class.

Step 2 (Creating ontology): Creating ontology involves creation of class-subclass relationship and attaching slots with the classes. For creating class-subclass hierarchy, class TAB from the toolbar needs to be selected. In a newly created project, there are only SYSTEM classes in the class pane. System class THING: needs to be selected as parent for any root level classes of taxonomy. For example, in creating student class (as the root of the given taxonomy) first, THING: needs to be selected as parent class, then ‘C’ button from the class button or “create new class” option from the popup menu needs to be activated. Similarly, graduate and undergraduate classes can be created (figure-2.12) from the Student class. After creating these subclasses, further subclasses can be created in the similar way. In ontology, classes are defined in terms of slots, which are attributes of the class. Slots can be created and attached with specific class (Template slots that indicated in figure 2.12). When a slot is created, it becomes a global object and can be attached to any class in the hierarchy, for this reason duplicate slot name is not allowed in a project. Slots are defined in terms of their facets, which can be specified based on the slot-template displayed in the right side window of figure 2.13.

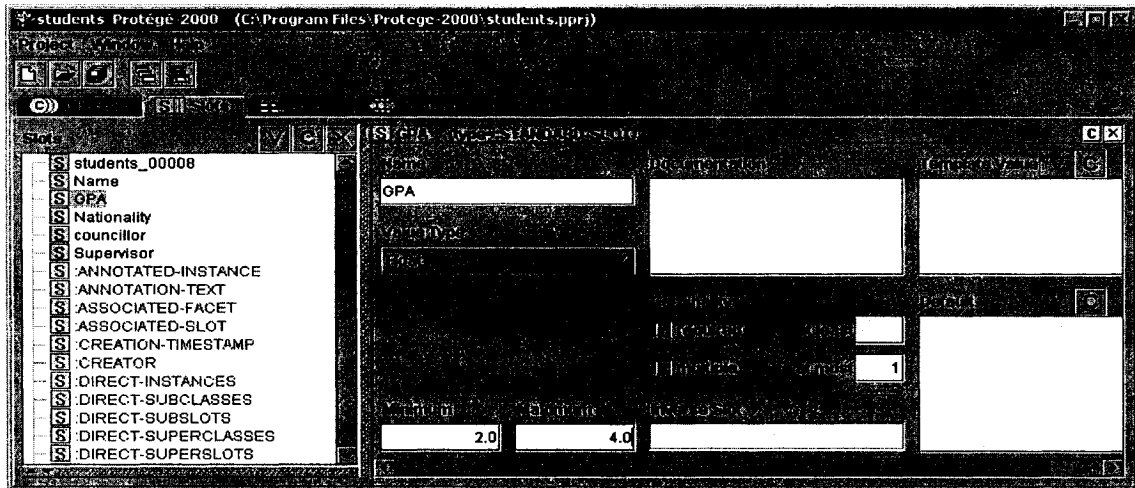


Figure 2.13: Slot tabbed window for creating and modifying slots.

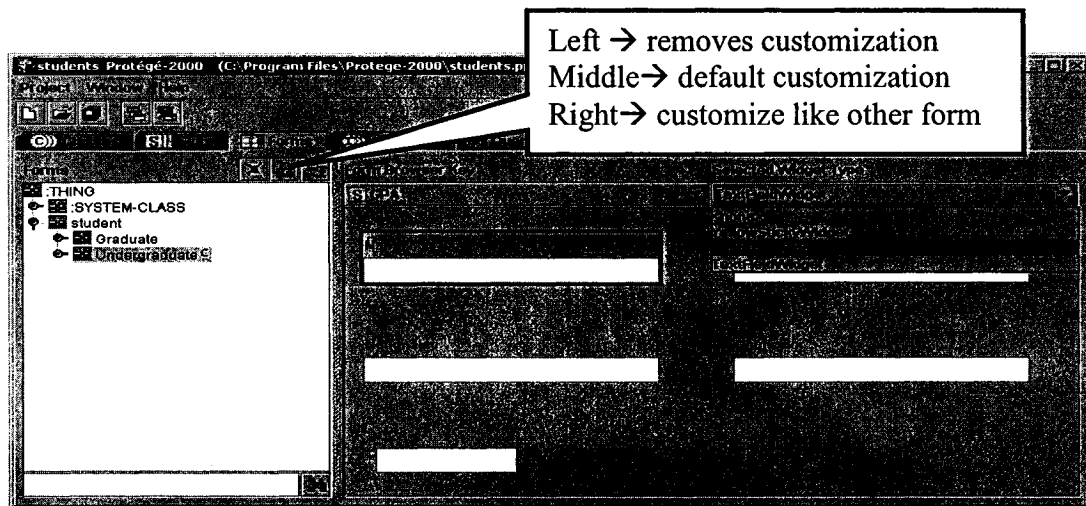


Figure 2.14: Form-tabbed window for customization.

Step 3 (Customizing the Forms): Customization of forms involves reorganizing the forms in such a way that users can insert and browse instances in the knowledge bases easily. For customization, form tab is selected, which consists of class pane and form-editing pane. Class pane contains the list of classes in the ontology, and form-editing pane displays the form corresponding to the selected class in the class pane. In the form, widgets are associated with the slots in the Class. Individual form is generated for each

class automatically. For ease of use, forms can be changed like other class form or it can be updated manually by dragging and dropping the widgets. Options are also available to select particular widget format for a selected widget from the widget menu. Figure 2.14 shows the selected widget that can take any of the formats from the given pull down menu.

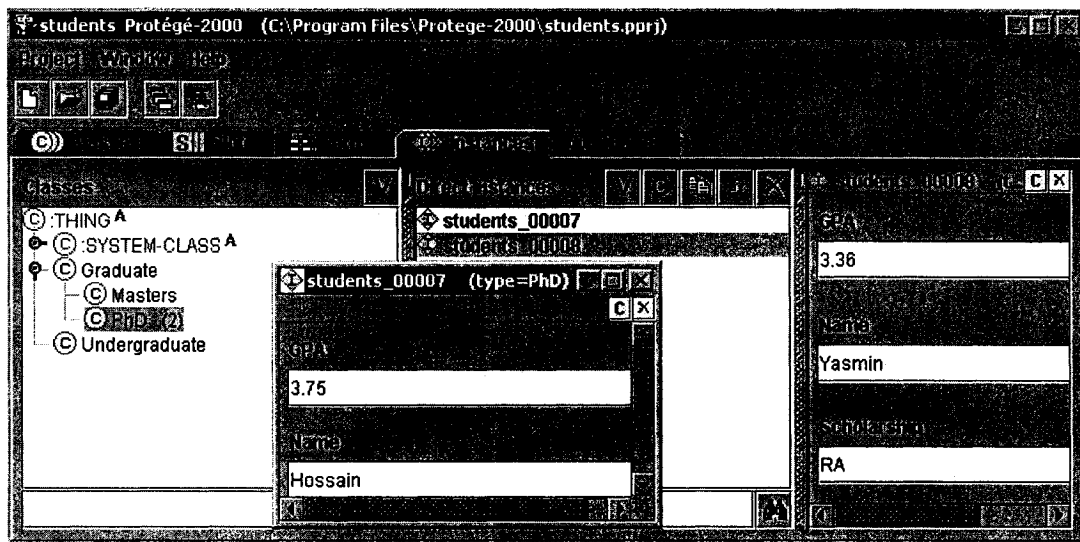


Figure 2.15: Instance tab for browsing and inserting Instances.

Step 4 (Entering and Browsing Instances): Inserting and browsing instances are most important activities in a knowledge-based system. Instance tab consists of three panes: class-pane, direct instance pane, and instance browsing pane. Class pane is used to select particular class, where instance needs to be inserted or browsed. Direct instance pane provides way for creating instance. Instance can be added by pressing “C” button in the direct instance pane. Instance-browsing pane provides way for browsing and inserting instances. After creating an instance, information can be added as form fill-up way in the instance-browsing pane. Selecting the browsing key from the direct instance pane (figure 2.15) can browse the instances already stored in the knowledge base

Step 5 (Create a query): Two different types of querying are possible, automated query generated by machine in machine learnt expert systems and user-generated query to obtain particular results from the database. Query tab, which is not part of the basic Protégé integrated system can be added to the system to create query. Figure 2.16 shows a query in the query box and the results displayed in the search results pane. The generated query can be added to the query library for further use.

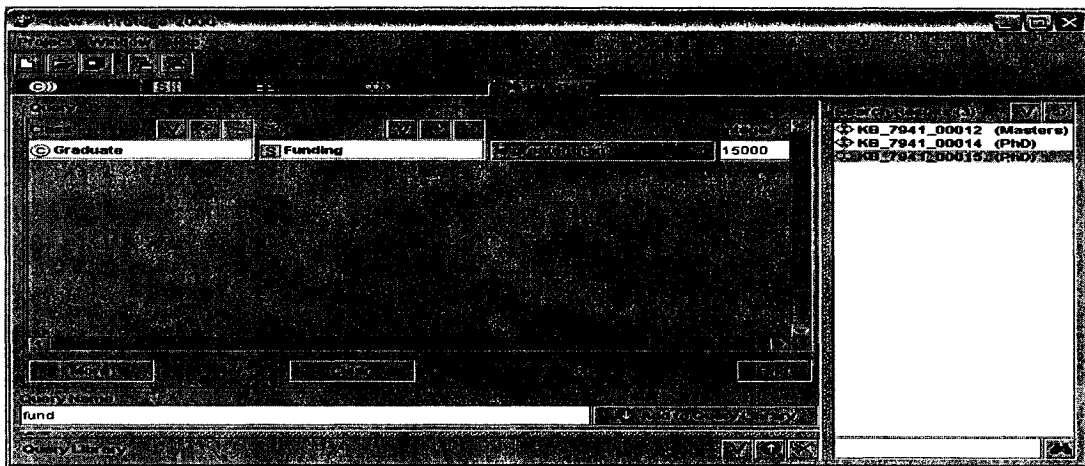


Figure 2.16: Query generating tab.

Protégé 2000 KA tool creates knowledge base in two steps. First, it acquires domain conceptualization (concepts, attributes and relationship) to create the ontology, which used as the structure of the knowledge base. Later, domain instances are acquired. A computer program can use the knowledge base for solving problem. Also, query can be created manually to browse the knowledge base instances. For example, I created a query by using the query plug-in, which is not part of basic Protégé system, to browse the created knowledge base and it is shown in Appendix B3.

2.3. Summary

In this chapter, Protégé 2000 and PCPACK4 have been described from the users' point of view. The following are the comparative summaries of these two knowledge acquisition tools Protégé and PCPACK:

1. Protégé and PCPACK both provide a graphical user interface for modeling domain concepts, attributes, and their relationship.
2. Protégé and PCPACK are similar in structuring the knowledgebase; both use ontology as the base of the knowledgebase.
3. The resulting PCPACK and Protégé knowledge base can be stored in different formats, such as XML, ASCII, RDF, and a relational database. In addition, the desired format can also be added to the Protégé system through plug-ins.
4. Protégé has wide varieties of users, such as system developers, knowledge engineers, and domain experts. System developers create new plug-ins via Java API to augment functionality to the Protégé open framework. Knowledge engineers and domain experts can build conceptual models of the domain, and can customize the forms for inserting and browsing instances.
5. PCPACK is also aimed at developing knowledge-based Internet sites for providing knowledge-based services through the Internet.

3. Techniques and Methods of Usability: An Overview

3.1 What is Usability?

Usability is the capability of a product that enables the users to learn the product quickly and use it easily to accomplish their own tasks (Dumas and Redish, 1994). Usability aspects are integrated into the product throughout the system development cycle in a systematic approach known as “usability engineering”. This includes identifying users, analyzing tasks, setting specifications, developing and evaluating prototypes, and the iterative cycles of development and evaluation (Dumas and Redish, 1994). Usability evaluation is performed in different stages of the product to provide feedback in the iterative system development (Gould and Lewis, 1985). Evaluating prototypes or systems involves measuring the usability of the product and identifying the user-interface problems [Dix et. al., 1998; Nielsen, 1993].

Usability is measured in terms of several usability attributes to examine the capability of the product. Although there is a consensus about the concept of usability, there are diverse approaches [Shackel, 1991][Nielsen, 1993][ISO 9241-11][ISO 9126-2] to define usability in terms of usability measures. For this thesis, we adopted ISO 9126-2 definition that deals with certain capabilities of the software product: (1) understandability, (2) learnability, (3) operability, and (4) attractiveness to the user when used under specified conditions. ISO 9126-2 defined a set of metrics to measure understandability, learnability, and operability. We assume that attractiveness can be measured by a set of subjective satisfaction questionnaire to measure the degree of satisfaction.

3.2. Usability Measures

We consider that the primary end-users of Protégé and PCPACK KA tools to be the domain experts themselves. Therefore, the success of these tools depends on the usage of these tools by such end-users. Considering this, the following measures were selected for the experimental study of this thesis:

1. Learnability – Objective measure.
2. Understandability- Objective measure.
3. Satisfaction- Subjective measure.

Understandability is defined as the capability of a software product to enable the user to understand: (1) whether the software is suitable for a particular purpose, (2) how it can be used for particular tasks under particular conditions (ISO 9126-2). Understandability plays an important role to motivate the end-users in using the product. For this reason, measuring understandability is important for the test of Protégé and PCPACK KA tools where the major target users are end-users who have little computer experience. This attribute will depend on many factors including documentation and initial impressions given by the software. For measuring the understandability, ISO 9126-2 defined the following metrics:

- (1) What proportion of functions (or types of function) can be identified based upon start up conditions? **
- (2) What proportion of interface functions is understandable? **
- (3) What proportion of the demonstrations / tutorials the user can access?

** Selected metrics for measuring understandability and learnability.

- (4) Can users understand what is required as input data and what is provided as output by software system?

The proposed test measures the understandability of Protégé and PCPACK KA tools for domain experts who have minimal degree of computer experience. Accessibility (first metric) and functional understandability (second metric) are two important understandability indicators for novice users. For this reason, we selected the first two of the above metrics for measuring the understandability of Protégé and PCPACK KA tools. The remaining (third and fourth) metrics can suitably be measured through the redesigning of the experiment, which is costly in the scope of this thesis. For this reason, they were not measured in the designed test.

Learnability is the capability of the software product to enable the user to learn its application. Learnability is important for systems where users of the system are intermittent rather than regular (Nielsen, p.80, 1993). For this reason, measuring learnability is important for Protégé and PCPACK KA tools' end-users who use these tools intermittently and have little computer experience. ISO 9126-2 defined the following metrics to measure learnability:

- (1) How long users take to learn how to use a particular function?
- (2) The proportion of the help topics the user can locate **
- (3) The effectiveness of help system and documentation. **

For measuring the first metric, users need to keep trying until the selected tasks are successfully performed. On the other hand, locating required help options and able to use

them efficiently help end-users in learning the new functionalities of Protégé and PCPACK KA tools. For this reason, the learnability of Protégé and PCPACK KA tools was measured through the second and third metrics. The selected measures indicate the degree of support that help-system can provide in learning and using the tools

Satisfaction is the degree of pleasantness felt when the users are using the product. Measuring satisfaction is a subjective process generally measured through a set of subjective questionnaire. For the test of Protégé and PCPACK KA tools two types of questionnaires were used. First, every user was asked: (1) Are any parts of the interface confusing and difficult to understand? (2) What is the worst aspect of the interface? The second part of the questionnaire was designed to obtain the satisfaction ratings on the following aspects of both knowledge acquisition tools:

1. Starting the tool
2. Creating ontology/ladder
3. Customizing form/ annotating the tool
4. Inserting instances
5. Selecting icon from the toolbar
6. Using Menu bar and popup menu
7. Understanding naming/ labeling
8. Overall

Table 3.1: Satisfaction questionnaire for Protégé and PCAPCK KA tools.

3.3. Usability Evaluation

Usability evaluation is concerned with collecting data about the usability of a design or a product by a group of users performing particular activities within a specified test environment or work context [Preece, J. et al., 1994 p. 602]. Many techniques and methods [Table 3.2] have been developed to evaluate usability in different stages of software lifecycle over the last 15 years. Based on the way of conducting the test, [Zang] categorized these methods into three different categories:

1. Inspection
2. Inquiry and
3. Experimental/empirical

METHODS	Requirement.	Design	Code	Test	Deployment
Proactive field study	✓				
Pluralistic walkthrough		✓			
Teaching method		✓	✓	✓	
Shadowing method		✓	✓	✓	
Co-discovery learning		✓	✓	✓	
Question asking protocol		✓	✓	✓	
Scenario based checklist		✓	✓	✓	✓
Heuristic evaluation		✓	✓	✓	✓
Thinking aloud protocol		✓	✓	✓	✓
Cognitive walkthroughs		✓	✓	✓	✓
Coaching method		✓	✓	✓	✓
Performance measurement		✓	✓	✓	✓
Interview		✓	✓	✓	✓
Retrospective testing		✓	✓	✓	✓
Remote testing		✓	✓	✓	✓
Feature inspection			✓	✓	✓
Focus groups				✓	✓
Questionnaires				✓	✓
Field observation				✓	✓
Actual logging				✓	✓

Table 3.2: Overview of Usability Evaluation methods (Folmer, E. 2004).

3.3.1. Usability Inspection

Usability inspection methods are evaluation methods involving usability experts examining the software user interface. In this technique, usability specialists, software developers, users and other professionals examine and judge whether each element of a user interface or prototype follows the established usability principles. Most of the inspection methods are used early in the software lifecycle to discover the usability problems, and some others are used addressing overall system usability covering the final prototype.

The following are the brief descriptions of the main methods of this category.

1. Heuristic Evaluation (Nielsen, 1994) involves usability specialists who judge whether each dialogue element follows established usability principles.
2. Cognitive Walkthrough (Wharton et al 1994) and (Rowley et al 1992) uses a detail procedure to simulate task execution at each step through the dialogue, determining if the simulated user's goals and memory content can be assumed to lead to the next correct action.
3. Feature Inspection (Nielsen, 1994) lists sequences of features used to accomplish typical tasks, checks for long sequences, cumbersome steps, steps that would not be natural for a user to try, and steps that require extensive knowledge in order to assess a proposed feature set.
4. Pluralistic walkthrough (Bias, R., 1991) uses group meetings where users, developers, and usability experts step through a learning scenario, discussing each dialogue element.

5. Standards inspection (Wixon et al, 1994) during which experts inspect the interface for compliance with certain standards. This can involve user interface standards as well as domain-specific software standards, departmental standards if they exist, etc.

Among the above inspection methods, heuristic evaluation is one of the least expensive and popular methods used for testing from early design stage to functional prototype evaluation [Victoria, L. E., 1999]. Considering the importance, heuristic evaluation method is described in the following:

Heuristic Evaluation

Heuristic evaluation (Mack & Nielsen, 1994) is one of the most informal methods where a small set of evaluators find usability problems by checking them against a set of heuristics or principles. The most commonly used heuristics is a set of interface design principles collected by Jacob Nielsen (1994). Table 3.3 summarizes the basic guidelines used to conduct the inspection by usability experts.

Heuristic evaluation doesn't involve any end-users, for this reason it is fast, cheap, and easy, but the problems discovered from this method is not always significant [Kelly et.al, 1995]. Actually, heuristic evaluation is suitable if there are qualified usability experts in the organization; otherwise, it might produce misleading results. The heuristic evaluation can preferably be used early in the development process to reveal design problems. The evaluations can be conducted on early stage prototypes, including paper mockups, as well

as later-stage electronic prototypes, with or without all of the back-end functionality implemented.

HEURISTICS	EXPLANATION
1. Simple and natural dialogue.	Dialogues are achieved through text, graphic design and color. They should not contain rare and irrelevant information because it competes with the relevant units of information and diminishes their relative visibility.
2. Speak the users' language.	The dialogue should be expressed clearly in words, phrases, and concepts familiar to the users, rather than system oriented.
3. Minimize the users' memory load	The users should remember information from one part of the dialogue to another, and icons should be easy to recognizable.
4. Consistency.	Follow platform convention and standards.
5. Feedback.	System should always keep informed the users what going on through appropriate feedback within reasonable time.
6. Clearly marked exit.	Users often chose system functions by mistake and will need a clearly marked emergency exit to leave the unwanted state.
7. Shortcuts.	Speed up the interaction for the expert users mostly unseen by the novice users.
8. Good error messages.	Plain errors message easy to understand for recovering from problems.
9. Prevent errors.	Prevents a problem from occurring.
10. Help and documentation.	Easy to search and find information.

Table 3.3: Definitions of heuristics (From Nielsen, 1994).

3.3.2. Usability Inquiry

Usability inquiry requires usability evaluators to obtain information about users' likes, dislikes, needs and understanding of the system by talking to them, observing them using

the system in real work (not for the purpose of usability testing) or letting them answer questions verbally or in written form. Inquiry methods include:

- Field observation (Nielsen, 1993)
- Interviews/ Focus groups (Nielsen, 1993)
- Survey (Alreck and Settle, 1994)
- Logging (Nielsen, 1993)
- Questionnaire (i.e. SUMI, WAMMI (HFRG)).

For evaluating usability of desktop and web based application, Questionnaire is one of the widely used methods (Folmer et al., 2004). Software Usability Measurement Inventory (SUMI) [Kirakowski et al. 1993] is one such questionnaire collection used for assessing satisfaction measures of the application. SUMI is concerned with the perception and feelings that the user has when working with a piece of software. The principle behind this questionnaire is that user perceptions are very important and must be taken into account in evaluating any software (Kirakowski & Cobertt, 1993)

SUMI is internationally standardized 50-item questionnaire. It takes a maximum of 10 minutes to complete and could work with small user sample sizes (Kirakowski & Cobertt, 1993). SUMI results have been shown reliable, and used to compare different kinds of software products. SUMI results are analysed into 5- sub-scales: affect, efficiency, helpfulness, control, and learnability. These scales present a view of subjective usability for which there is high level of empirical support.

3.3.3. Empirical Testing

The usability testing approach requires representative users to work on typical tasks using the system or a prototype. A Prototype models the final product that allows testing of the attributes of the final product even it is not ready yet. The evaluators use the results to see how the user interface supports the users to do their tasks. Empirical testing includes the following methods:

- Coaching Method (Nielsen, 1993)
- Co-discovery learning (Nielsen, 1993), (Dumas and Redish 1993)
- Performance Measurement (Nielsen, 1993)
- Question asking protocol (Dumas and Redish 1993)
- Remote Testing (Hartson et al., 1996)
- Thinking aloud method (Nielsen, 1993)

Most of the above methods are used to collect qualitative data except Performance measurement method, which is used to collect quantitative data. Thinking aloud method is the most used method used to collect qualitative usability problems. For the test of Protégé and PCPACK qualitative and quantitative data needs to be collected. Considering this, two most used methods are described in the following:

Performance Measurement Method

Performance is almost always measured by having a group of test users perform a predefined set of test tasks while measuring task time and other related data (Nielsen J., p-192, 1993).

Performance measurement starts with an abstract concept, such as measuring the usability. To obtain such type of a goal one needs to break this goal into some components (i.e. learnability, understandability) for having the expected results. Once the components of the goal have been defined, it becomes necessary to quantify them precisely. For example, the component “learnability” needs to be quantified. For this, a number of metrics have been selected that already been defined ISO 9126-2. Given the quantification of a metrics, one needs to define a method for measuring the metrics. There are two obvious alternative ways. First one, bring some test users into the laboratory and give them a list of tasks to perform, second one is to observe a group of users at work in their own environment and measure them whenever a task like the specified test tasks occur. For the first method, a systematic approach to perform the test and compute the results is necessary. A major pitfall with respect to such measurement is the potential for measuring something that is poorly related to the property one is really interested in assessing [Nielsen, J., 1994]. For example, users might be interested to measure the overall learnability but the method only provides help accessibility and help frequency and learning time for specified tasks.

Thinking Aloud Method

Thinking aloud test involves having a test user using the system while continuously speaking aloud what s/he is thinking (Lewis 1982). By verbalizing their thoughts, the test users enable the evaluators to understand how they view the computer system, and this again makes it easy to identify the users’ major misconceptions. One gets a very direct understanding of what part of the dialogue causes the most problems.

The main disadvantage of this method is that it does not lend itself very well to most types of performance measurement. On the contrary, its strength is the wealth of qualitative data it can collect from a fairly small number of users. At the same time, thinking aloud also gives a false impression of the cause of usability problems if too much weight is given to the users' own theories of what caused trouble and what would help. The most difficult part in thinking-aloud is verbalizing because user performs many operations so quickly, but they may have nothing to say. They may not even consciously know what they are doing in cases where they have completely automated certain common proc

3.4. Choosing Usability Technique and Methods

The kind of evaluation technique and methods required for a test depends on the purposes of the test. The purpose of the tests in our case is to assess the usability of KA tools against a set of criteria, and to provide recommendations for further improvement of these tools or to help in the development of new KA tools. These objectives can be achieved through the performance of a set of test tasks by representative end-users. Therefore, user-based experimental testing is the only option for this study.

There are many aspects of usability that have to be considered in selecting an experimental method for a particular test. Every method has its own strength, weakness, and used for specific context. No single method is sufficient enough for measuring usability in all the stages of the software development lifecycle [Jacobsen, 1999],

therefore, methods needs to be selected considering the every individual context, for this reason, often more than one method chosen for the usability testing. The methods for a particular context can be selected based on the following issues [Dix A., 1998]

- The stage in the cycle at which the evaluation is carried out i.e. either at design or implementation stage.
- The required level of subjectivity or objectivity from the test
- Type of measures needed either qualitative or quantitative
- Resources available for the method
- Information provided by the method
- Usefulness of the method in the context of use.

After a thorough analysis the experimental methods with respect to above issues, laboratory based performance measurement can be selected for this test. Performance measurement method in the laboratory used to collect performance data. Along with performance data, problems faced by participants during the test session can be obtained through the review of the participants' activities. It is difficult to measure satisfaction from the task performances. For this reason, I have created satisfaction questionnaire to collect satisfaction scores on different aspects of Protégé and PCPACK KA tools. The subjective questionnaire was administrated at the end of the test session. Once the technique and methods are selected for the study, it is important to devise a systematic method to conduct the study. We believe there is no systematic approach to conduct the experimental study of KA tools. For this thesis, we adopted "Checklist based methodology" [Hix et al. 1991] that was not designed for KA tools but is intended for

general interactive systems. In chapter 4, we follow this methodology and make certain minor modifications in it to apply it to the study of the two selected KA tools.

4. Usability Testing of Knowledge Acquisition Tools: A New Approach

Knowledge Acquisition tools belong to a class of interactive systems that provides an environment for developing knowledge bases. Despite recent proliferation of such tools, it is believed that there are no procedures for systematically evaluating the usability of KA tools. The closest tool evaluation research was the “Checklist-based methodology (Hix, et. al. 1991) for evaluating and comparing human computer interface development tools.” This research was not specialized to KA tools but is general for all interactive systems. We follow this methodology and make certain minor modifications and apply it to the study of two selected KA tools.

4.1. Hix’s Checklist Based Methodology

In 1983 Robert and Moran produced a methodology for evaluating text editors. They considered a common basis for all editors along four dimensions:

- a) Time to perform tasks.
- b) Errors and cost of errors.
- c) Learning time.
- d) Tool’s functionality support.

Based on the above research, 8 years later Hix and Schulman developed a checklist based methodology for evaluating User Interface (UI) development tools. The following are the major characteristics of Hix’s methodology:

- It is based on 28-page “checklist” matrix built along two dimensions: functionality and usability. This list is called the evaluation form. Functionality indicates what the tool can do; that is, what interface styles, techniques, and features can be produced by the tool for a target application interface. Usability indicates how well the tool performs its possible functions, in terms of ease-of-use (a subjective, but quantitative rating such as: difficult (1), adequate (2) and easy (3)).
- The purposes of this methodology are to evaluate and compare human-computer interaction tools.
- It is an inspection-based evaluation approach where only well-trained UI expert participated in the study. The number of participants typically less than 10.
- They have also evaluated the validity of the methodology. Empirical study conducted by six evaluators was used to show the consistency of the results when performed repeatedly.
- In this methodology functionality rating is expressed as a percentage of the number of functions possible with the tool with respect to the total number of functions on the “evaluation form”. The usability is expressed as the percentage of the ease-of-use rating of all functions possible within the tool with respect to the maximum ease-of-use rating for those same possible functions.

Based on the above characteristics, Hix’s methodology used the following steps in evaluating user interface development tool:

Step 1: Acquire user interface development tools to be evaluated and compared.

Step 2: Learn and use the tools being evaluated. Three stages are suggested in learning the tools and using the methodology.

- Users will learn how to use the tool(s) to a reasonable level of expertise.
- Users will perform baseline tasks for the experimenter, to ensure that participants attained a minimum level of expertise in using the tool(s).
- Users will be given training on the use of the form and glossary by the experimenter.

Step 3: Complete the evaluation forms for each tool.

Step 4: Perform the benchmark tasks (optional)*.

Step 5: Compute functionality and usability.

4.2. Our Objectives

The study of Protégé and PCPACK KA tools is based on the following objectives:

- We wish to conduct an experimental study.
- The two purposes of this study are: (1) to measure usability attributes in terms of three factors: a.) Learnability, the capability of the product that enables users to learn its application, b.) Understandability, the capability of the product that enables users to understand its suitability for KA purpose, and c.) Satisfaction, the degree of users' satisfaction; (2) to discover the problems, if any, with the two selected KA tools and suggest improvements for the future.

* This is a traditional benchmark testing using representative tasks that was not addressed in Hix's methodology.

- The participants in this experimental study are to be drawn from the student community in order to make the experiment viable for us.

4.3. Modified Approach to Evaluate Protégé and PCPACK KA Tools

Considering the objectives of the intended study, the steps used in Hix's methodology have been modified to result in the following six steps:

Step 1: Selection of the test or benchmark tasks and participants with regard to KA tools.

Step 2: Creation of the format of the tables to be used as "evaluation forms"

Step 3: Participants learning the tools.

Step 4: Participants perform the tasks.

Step 5: Review of the sessions and collect data using the table ("evaluation form").

This was done by the researcher.

Step 6: Compute metrics and interpret results.

The following are the description of the proposed six evaluation steps in detail:

4.3.1. Selection of the Test Tasks and Participants

Hix's evaluation methodology uses inspection-based technique where evaluators select possible functions from the total functions given on the "checklist" form. However, the experimental study of Protégé and PCPACK KA tools, which involves preparing laboratory and test materials, recruiting users of the target systems, and managing test sessions is an expensive and time consuming process. Considering the expenses and time constraint, instead of trying all the possible tasks, for the experiment we choose to use a

small number of representative test tasks those will provide reasonable coverage of the most important part of the user-interface. Similarly, a set of representative users need to be selected those will represent the real end-users of these tools.

4.3.1.1. Selection of Test Tasks

The test tasks can be selected based on user-defined tasks or based on product-supported tasks [Corde, R.E., 2001]. User defined tasks are based on user requirements and they are selected by having users bring into the laboratory certain tasks they want to perform and believe they should be able to perform with the product [Corde, R.E., 2001]. On the contrary, product supported tasks are based on the intended uses of the systems, which can be found in the identity statement listing for the product. Ruben (1993) suggested to select the product supported test tasks based on frequency (most frequently performed tasks), criticality (if performed incorrectly or missed, have serious consequences to the users) and vulnerability (suspected tasks that might be hard to use or that have known design flaws) of the tasks. Also, information from the logging of frequencies of use of commands in running systems and other ways of learning how users actually use system such as field observations can be used to construct representative sets of test tasks for UI evaluation (Gaylin 1986). For user-defined tasks, users should be familiar with the intended activities of the target system or use some earlier versions of the product. However, the intended users of this test are domain experts who have very little degree of experience in computer and have no experience in knowledge acquisition tools. For this reason, the selection of user-defined tasks for our test is not suitable. For our test, we selected product-supported tasks based on the intended uses of the systems. The intended

purpose of Protégé and PCPACK KA tools is to create and maintain knowledge base. Based on this purpose, for the experiment with Protégé and PCPACK KA tools, we selected the following major task:

“Develop a knowledge base for maintaining student information”.

This type of major task is also known as goal or desired state of the system. However, (Nielsen, 1993, p.185) tasks should be small enough so that they can be completed within the time limits of the user test, but they should not be so small that they become trivial. Since, the selected task is large, a set of sub-tasks can be derived from the large task using task analysis, which is aimed at splitting any work activity into its basic elements and each being directed to a particular goal (Clegg. et al. 1988, p.168). Hierarchical Task Analysis (HTA), described by Annet, Duncan, Stammers and Gray (1971) is one of the popular action oriented techniques for describing how work is organized in order to meet the overall goal and it was used for this purpose. Among many task analysis methods, the selection of HTA for this purpose was based on its strength in breaking the large-task into a number of sub-tasks. The following brief overview of the task analysis methods will justify the selection of the HTA over other methods:

The task analysis techniques may be grouped together in many ways, but probably the most important distinction is whether the technique aims to represent cognition, practice or logic of the task (Payne and Green, 89). The techniques that aimed to represent cognition focus on the mental processes, which underlie observable behavior (e.g. decision making and problem solving), are referred to as cognitive approaches. Cognitive task analysis techniques seek to model the internal representation of knowledge that

people have or need to have in order to complete a task (Preece, 1994). Among many cognitive task analysis techniques, GOMS (Goals, Operations, Methods and Selection rules) (Card et al. 1983) used for modeling how-to-do-it knowledge and, KAT (Knowledge Analysis of Tasks) (Johnson, 1992) used for identifying previous knowledge of users relevant to the generic and specific tasks. On the contrary, practice oriented techniques provide a minimum description of the observable aspects of the operator behavior of various levels of details together with some indications of the structure of the task (Embrey, 2000). Decision flow diagram and Hierarchical Task Analysis are the commonly used [Embrey, D., 2000] practice oriented methods. Decision flow diagrams can be used to represent tasks, which involve decision-making, time-sharing, and it also can be used to identify critical checks that the users have to carry out. However, HTA describes how work is organized, in order to meet the overall goal which involves an iterative process of identifying tasks, categorizing them, breaking them down into subtasks and checking the accuracy of the decomposition (Preece, 1994, p.413). These imply that HTA conform to the intended purpose to derive the task list from the specified goal (development of a KB for student information system).

In HTA, most important question involves deciding upon the level of detail that is required at what point to stop the decomposition. This can range from a fine level of description, in which individual keystrokes (e.g. click mouse) are outlined to a higher level, in which basic units of activity are described (e.g. delete a class) (Preece, 1994, p.416). It depends on the analyst who is performing task analysis (Embrey, 2000). For design purpose, tasks are decomposed into keystroke level and describe the details in

terms of operation and plan that consists of a number of task-actions. However, for deriving the tasks to conduct user studies the large task (goal) can be decomposed into a number of higher-level tasks. The following figure 4.1 shows how tasks are decomposed into a number of sub-tasks.

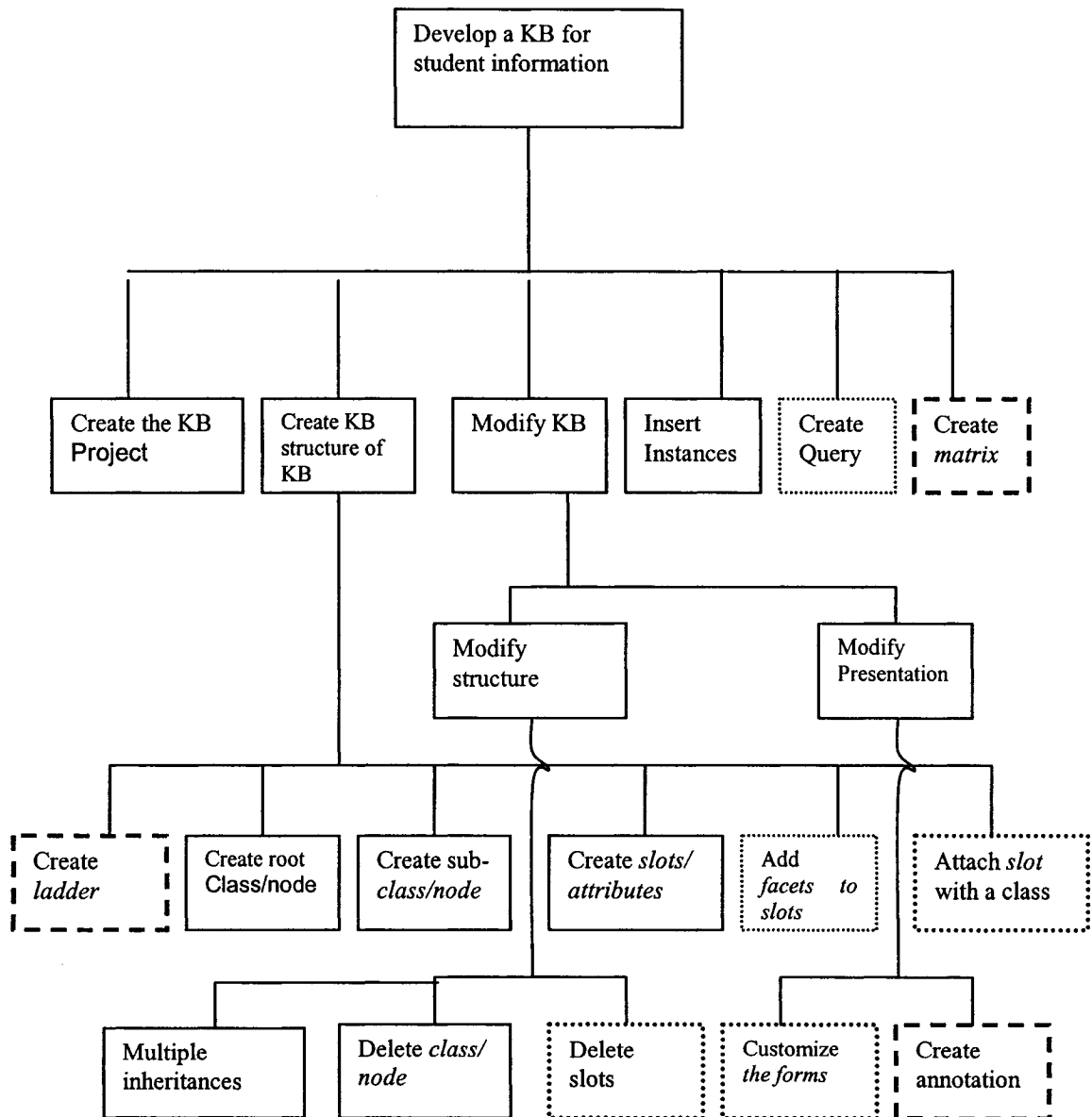


Fig 4.1: Hierarchical task analysis to derive the test tasks (leaf node) [□ = Common tasks, [-] = PCPACK task only, [···] = Protégé task only].

Name	Description
<i>Project</i>	A project is a file that contains the presentation information and references to external sources of the domain information.
<i>Ladder</i>	A ladder is a hierarchical (tree-like) network diagram.
<i>Class</i>	Representation of a concept in a domain.
<i>Node</i>	Representation of a concept in a domain.
<i>Slot</i>	An attribute of a class.
<i>Attributes</i>	An attribute is a quality or feature of PCPACK object.
<i>Facets</i>	The attributes of a slot.
<i>Forms</i>	The Protégé-2000 use forms for acquiring instances of classes.

Table 4.1: Descriptions of technical terms used in figure 4.1.

The leaf nodes in figure 4.1 are test tasks. There are three types of leaf-nodes. Solid-boundary nodes are common tasks for both Protégé and PCPACK KA tools, star-boundary nodes are only for Protégé KA tool, and broken-line boundary nodes are only for PCPACK KA tool. Although Protégé and PCPACK tools use ontology for structuring the knowledgebase, the differences in their features and functionalities create differences in performing the same large task. For this reason, a number of derived tasks that are possible in one tool are not possible in other. Table 4.2 shows the lists of test tasks that have been derived from the above task analysis.

The test of Protégé and PCPACK knowledge acquisition tools includes tasks to create knowledge base structure and to acquire instances in the knowledge bases. The browsing of knowledge bases, which includes creating and executing queries, were not included in this test because the basic Protégé and PCPACK system do not support these activities. For performing browsing, a query plug-in that is not part of the original system needs to

be added to the protégé system; and a browsing interface needs to be created for PCPACK knowledgebase. Adding new plug-ins and creating browsing interfaces are beyond the capabilities of end-users. For this reason, they were not included in this test.

TASKS	PROTÉGÉ	PCPPACK
1. Create a project	✓	✓
2. Create a new ladder	NA	✓
3. Create the root level class/node for the taxonomy	✓	✓
4. Create a subclass/children node for a class/node in the given taxonomy	✓	✓
5. Create a slot/attribute	✓	✓
6. Specify the artifacts of a slot	✓	NA
7. Attach a slot with a class	✓	NA
8. Create multiple inheritance for a class/node	✓	✓
9. Delete classes/node	✓	✓
10. Delete slots	✓	NA
11. Create the annotation template for a node	NA	✓
12. Customize a form	✓	NA
13. Insert an instance to the knowledge base	✓	✓
14. Create a query	✓	NA
15. Create attribute matrix with the matrix tool	NA	✓

Table 4.2: Task List for Protégé 2000 and PCPACK4. (Reformatted from figure 4.1).

4.3.1.2. Test Participants

Test participants should be as representative as possible of the intended users of the system (Nielsen, 1993 p.175). The target audiences for KA tools would be domain experts and knowledge engineers. Since, our aim is to investigate how domain experts perform with these tools, representative domain experts have been selected for the test. Protégé 2000 and PCPACK4 KA tools' users generally would be from different age

group, sexes, cultures, and backgrounds, but it is difficult for us to recruit users from each category. Considering the task domain, we have recruited ten graduate and undergraduate students from different educational and cultural backgrounds.

Recruited participants had no prior experience with these tools. Three of the ten test participants were computer science undergraduate first year student. They were between the ages of 20 to 25. They had no work experience, but were familiar with programming as well as a number of other application software. The others were Electrical Engineering, Civil Engineering, and Mathematics and Statistics graduate students who had work experience in their area for at least five years. Two of the ten users were native English speakers and others were non-native language speakers but had good command in reading, writing and listening.

4.3.2. Creation of the Table

Hix's methodology uses a 28-page fixed "checklist" forms to record possible functions and their corresponding ease-of-use scores, which in turn used to measure functionality and usability. The purposes of this experimental study are to collect usability problems and to obtain usability measures (e.g. learnability, understandability, and satisfaction). In this study, usability problems can be identified by reviewing the recorded test sessions; satisfaction in different aspects of a tool can be measured through a set of post-test questionnaire. For obtaining learnability and understandability metrics, the researchers need to plan something in advance so that later s/he can compute the metrics efficiently. Similar to Hix's "checklist" form, a table can be created along the two dimensions:

simple tasks and usability criteria to measure the desired understandability and learnability metrics. The simple tasks are task actions (functions) that obtained through the decomposition of the given tasks [Telford, 1999]. Based on this, the following understandability and learnability metrics have been adopted as the usability criteria, to form the evaluation table.

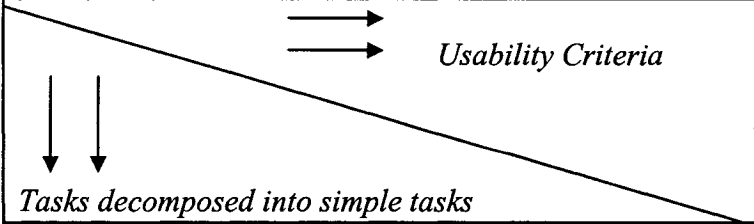
Column 1: Able to locate: Whether a user tried to locate the simple task in the start-up condition? It includes both able to perform and not able to perform a simple action. There are cases where the tool user cannot even locate task action.

Column 2: Able to perform without help: Whether a user able to understand and perform the located simple task?

Column3: Able to locate the help: Whether users able to locate the required help option in the help system? A user unable to perform a simple task looks for help in the help system and s/he may or may not be able to locate the help facility of the tool.

Column4: Able to use the help: Whether a user is able to understand and use the located help to perform the located simple task action?

Based on the above usability criteria and selected test tasks the following tables [Table 4.3 and Table 4.4] can be created for Protégé and PCPACK KA tools:

<div style="text-align: center;">  <p><i>Tasks decomposed into simple tasks</i></p> <p><i>Usability Criteria</i></p> </div>	Able to Locate	Able to perform without help	Able to locate the help	Able to use the help
Task1: Create a project				
Start the tool from the start-up menu				
Use Save the option				
Giving file name				
Locate the project destination				

Task2: Create the root level class for the taxonomy				
Select the parent Class (THING) for creating the class				
Select the option to create the class				
Name the class				
Task 3: Create a subclass for a class in the given taxonomy				
Select the parent class of the subclass				
Select the command				
Task4: Create multiple inheritance for a class				
Select the class for Multiple inheritance				
Select the super class pane				
Select another super class for multiple inheritance				
Task5: Delete classes				
Select the class to be deleted				
Select the option to delete the selected class				
Task 6: Create slot				
Select the class				
Select the button for creating template slot				
Name the slot				
Task 7: Specify the artifacts of a slot				
Select the type of the slot				
Specify the cardinality				
Specify maximum, minimum (if the type is integer)				
Specify template value				
Specify Default value				
Specify inverse slot (if applicable)				
Task 8: Attach a slot with a class				
Select the class				
Use the command to add the slot at class level				
Select the slot				
Task 9: Delete slots				
Select the specific slot				
Use menu option to delete the slot				
Delete the slot				
Task 10: Customize a form				
Select the proper browsing key				
Select the form customization				
Select the widget				
Move the widgets in the suitable place				
Select the suitable widget type				
Task 11: Insert an instance to the knowledge base				
Select the class for which instance need to be inserted				
Select and press the button to insert an instance				
Insert the values in the corresponding slots for that class				
Task 12: Create a query				
Select the class				
Select the slot in the class				
Select the logical condition				
Set the comparison value				
Execute the query				

Table 4.3: Evaluation table for Protégé KA tool.

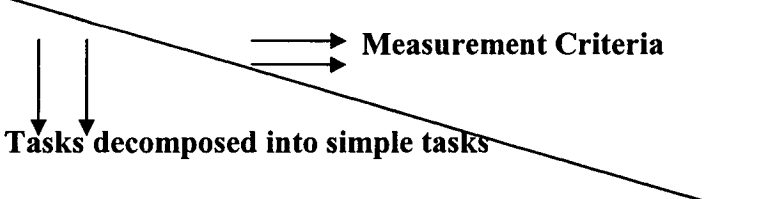
 Tasks decomposed into simple tasks	Able to Locate	Able to perform	Able to locate the help	Able to use the help
Task1: Create a project with ladder tool				
Start the ladder tool				
Select the menu option/toolbar icon to create knowledgebase				
Give name and password for knowledgebase				
Select the ontology for the knowledgebase				
Task2: Create a new ladder				
Locate the option/icon for creating new ladder				
Give name of the ladder				
Specify the relation				
Task 3: Create a root for the ladder				
Select the icon/ give command for creating a root.				
Name the root object				
Task 4: Create children for parent node in the ladder				
Select the icon/ menu command for creating child node				
Specify the relation between parent and child node				
Name the new child node				
Task 5: Create multiple inheritance for a node				
Select the second parent node				
Select the tangled command				
Select the command to create a child from second parent node				
Locate the child node that needs multiple inheritance				
Select the option to activate the child creation				
Task 6: Delete a node				
Select the object that has to be deleted				
Locate the command for deletion from Menu/icon				
Task 7: Assign the attributes to the concept				
Select the node for attaching attributes				
Select the command from popup menu to create attribute				
Give attribute name				
Task 8: Create the annotation template for a node				
Select the node				
Activate the annotation template				
Write the code for the template				
Task 9: Insert instances to the knowledge base				
Convert the leaf object node into instance				
Select the command to add attribute value in the instance				
Task 10: View the knowledge base with matrix tool.				
Select the command for creating attribute matrix				
To add/ remove attributes in X the axis				
To add/remove concepts in the Y-axis				
Add/remove attributes corresponding to the concepts in KB				

Table 4.4: Evaluation table for PCPACK KA tool.

4.3.3. Learn the Tools

In Hix's methodology, users (evaluators) need to select possible functions from the total functions on the "checklist" form. For this reason, they need to learn the detailed use of the tools. For reliable results, users need to attain a minimum level of expertise in using the tool. They suggested three stages of learning that have been mentioned before to ensure evaluators minimum level of expertise in using the tool(s). On the contrary, the test users of Protégé and PCPACK KA tools are not specialists and the purpose is to measure initial understandability, learnability and satisfaction in using these tools and to collect usability problems at that level. For this reason, its not needed to train the users like Hix's methodology, instead, before the test session, they will be given a short tutorial for one-hour to facilitate them to learn the tool by themselves.

4.3.4. Performing the Tasks

Hix's method is inspection-based, for this reason it didn't involve any task performance. In addition to the inspection-based subjective evaluation, Hix's method optionally suggested performing benchmark tasks to evaluate the user interfaces, but they did not address it within the given methodology. The study itself is an experimental test of the user interfaces, therefore; performance of test tasks is one of the important parts of the study.

This experimental study consists of two test sessions. In the first session, all users will be asked to perform a set of Protégé tasks; and in the second sessions they will be asked to perform a set of PCPACK tasks. Before the test, participants will be given a detailed task scenario (Appendix A4) rather than the task lists [Table 4.2] so that they can understand

what they have to perform with a particular KA tool. The task session will be captured through Camtasia screen capturing software. The software is also capable of recording the users' comments during the test. This approach has been developed by targeting the evaluation of the end-users' performance with KA tools: Protégé and PCPACK; for this reason, users will be allowed to take as much time as they need to complete a task so that they can do so without any time pressure. At the end of the session, users' satisfaction in using these tools will be measured subjectively through a number of post-test questionnaires (See: Appendix A5).

4.3.5. Review of the Sessions and Completing the Table Entries:

In Hix's methodology, evaluators completed a predefined "checklist" that in turn used to compute functionality and usability. In this study, Camtasia software recorded the task sessions with Protégé and PCPACK KA tools. For collecting required information, researchers reviewed the recorded sessions with the help of test participants so that they can help in answering any questions that arise during the review. During the review, the researchers looked for two types of data. First, usability problems those users faced during the test session. Second, user's interaction data corresponds to the simple task action (Table 4.3 and Table 4.4). During the review, researchers recorded the user interaction of a particular user in the evaluation table (Table 4.5). In this recording process, researchers checked the corresponding usability criteria for every simple task action. For instance, in performing the first simple task action of task1 a particular user able to locate it and performed it without any help. As a result, corresponding usability criteria (Able to locate, and able to perform without help) were checked in the given

evaluation form (Table 4.5). Similarly, in performing the first simple task of task2, the same user performed it with the usage of help system (i.e. the particular user was able to locate and use the help to perform the given simple task action). As a result, corresponding usability criteria (able to locate, able to locate the help, and able to use the located help) were checked for that simple task action. In this way, the researchers fill-up the evaluation tables for Protégé and for PCPACK KA tool. Table 4.5 is an example of individual evaluation table of Protégé KA tools. In terms of column table 4.5 is similar to the final table but in terms of rows it has fewer rows (simple tasks) in comparison to the final table.

		Measurement Criteria			
		Able to Locate	Able to perform without help	Able to locate the help	Able to use the help
Tasks decomposed into simple tasks					
Task1: Create a project					
	i. Start the tool from the start-up menu	x	x		
	ii. Use Save button (b)/Menu (m) option				
	iii. Giving file name				
	iv. Locate the project destination				
Task2: Create the root level class for the taxonomy		-	-	-	-
	i. Select the parent Class (THING) for creating the class	x		x	x
	ii. Select the command to create the class	x	X		
	iii. Name the class	x	X		
Task 3: Create a subclass for a class in the given taxonomy		-	-	-	-
	i. Select the parent class of the subclass	x	X		
	ii. Select the create command	x	X		

Table 4.5: Evaluation table of a particular user of Protégé KA tool.

4.3.6. Compute the Metrics

Hix's methodology computes functionality and usability based on the selection of possible functions with the tool and their corresponding ease-of-use rating. Functionality expressed as a percentage of the number of functions possible with the tool with respect to the total number of functions on the "evaluation form", and usability as the percentage of the ease-of-use rating of all functions possible within the tool with respect to the maximum ease-of-use rating for those same possible functions.

For the test of Protégé and PCPACK KA tools, usability metrics can be computed from the entries of the evaluation tables. For calculating usability metrics, researchers combined all the individual evaluation tables into a single table that they called "combined table". In this way, they created separate combined table for Protégé and PCPACK KA tools. In the combined table, the designated cell entries are no longer interaction entries, but they are the number of users' who performed a particular simple task successfully. In addition to the existing columns in the evaluation table, the combined table adds one extra column (first column). The combined tables for Protégé and PCPACK KA tools are presented in chapter 5. For example, a part of the combined Protégé table is presented in the following. In terms of usability criteria, the example [Table 4.6] of combined table is similar to final combined Protégé table presented in chapter 5, but different in terms of simple tasks, that is; it has fewer rows (simple tasks) in compare to final table. The content of these columns are described in the following:

1. First column provides the number of users who were available for a particular simple task. For a given test task, table 4.6 shows that the number of available

users in the first few simple tasks are almost 10, but the number of users decreases as the simple tasks proceed because a number of users face difficulties in the first few simple tasks, therefore; they abandoned the given task. For this reason, they were not available for the remaining simple tasks of a particular task.

2. The data in the second column indicates the number of users who were able to locate the particular simple task. This number can be less than or equal to the number of available users for the particular simple task.
3. Third column indicates the number of users who were able to perform the located simple task. It can be less than or equal to the number of users indicated in the second column.

Tasks decomposed into simple tasks		Measurement Criteria				
		Available users	Able to locate	Able to perform without help	Able to locate the help	Able to use help.
Task1: Create a project						
	i. Start the tool from the start-up menu	10	10	10		
	ii. Use Save button (b)/Menu (m) option	10	6	6		
	iii. Giving file name	6	6	6		
	iv. Locate the project destination	6	3	3		
Task2: Create the root level class for the taxonomy		-	-	-	-	-
	i. Select the parent Class (THING) for creating the class	10	10	7	2	2
	ii. Select the command to create the class	9	9	9		
	iii. Name the class	9	9	9		
Task 3: Create a subclass for a class in the given taxonomy.		-	-	-	-	-
	i. Select the parent class of the subclass	10	10	8		
	ii. Select the create command	8	8	8		

Table 4.6. Combined evaluation table of Protégé KA tool.

4. Fourth column indicates the number of users who were able to locate the required help. The users who were unable to perform the located simple task tried to locate help from the help system.
5. Fifth column indicates the number of users who were able to perform the simple task with the located help.

The following usability metrics can be calculated from the combined evaluation table[Table 4.6]:

- **Function Accessibility:** The ratio of users who were able to identify a function in the start-up condition to the number of users who were available to identify that function.
- **Function Understandability:** The ratio of users who were able to understand and use a function to the number of users who were trying to use the particular function.
- **Help Accessibility:** The ratio of users who were able to locate the help to the total number of users who wanted help. The users who located the function but unable to use it wanted help from the help system. This can be obtained by subtracting column3 from column2.
- **Help Efficiency:** The ratio of users who were able to understand and use the help to the total number of users who were able to locate that help.

4.4. Summary

Hix's methodology for the evaluation of user-interface development tool has been modified and applied to study Protégé and PCPACK KA tools from the end-users' point of view. In the modified approach, the important modifications include selection of test tasks for the experimental study and creation of evaluation tables to collect user-interaction data of Protégé and PCPACK KA tools. By reviewing the test sessions, we collected the usability problems, combined the evaluation tables to compute the usability metrics. The results of the experimental research are presented and interpreted in chapter

5. Analysis and Interpretation of Test Results

In this chapter, we take a closer look at the outcome of the usability study of knowledge acquisition tools. Section 5.1 presents the results obtained from the usability test. Section 5.2 provides an analysis of the problems collected from the study. Finally, based on the analysis, we make specific recommendations for the improvement of the two KA tools.

5.1. Results

The goal of this thesis has been to assess the usability of KA tools against a set of criteria, and to provide recommendations for further improvement of these tools or to help in the development of new KA tools. In order to fulfill this goal, we must understand not just what test participants did during the test session, but why they behaved and reacted as they did. This can be accomplished first by characterizing the test participants, and then by examining their performances, the satisfaction ratings, and the observed problems. In the following, the results obtained from the usability tests are presented in the same order, as they were collected.

5.1.1. Participant's Expertise

The participants were graduate and undergraduate students who had no prior experience with these tools. In total 10 participants were selected. Three of the ten test participants were Computer Science undergraduate students who were in their first year. The others were Electrical Engineering, Civil Engineering, and Mathematics and Statistics graduate students who had work experience in the area of their expertise. Two of the ten users were native English speakers and the others were non-native English speakers who had

good command in reading, writing, and listening English. The histogram in Figure 5.1 summarizes the usages of default computer software by the test subjects. It describes the number of users who were familiar with the software (Windows, Linux, MSword, Excel, Programming, and other tools) to a certain level of expertise. The histogram shows that all the users were familiar with the Windows operating system, and with Microsoft Word, and that seven users were familiar with Excel software. A number of graduate students had experience with other professional tools, such as MATLAB, ORCAD, and Computer Aided Drawing (CAD).

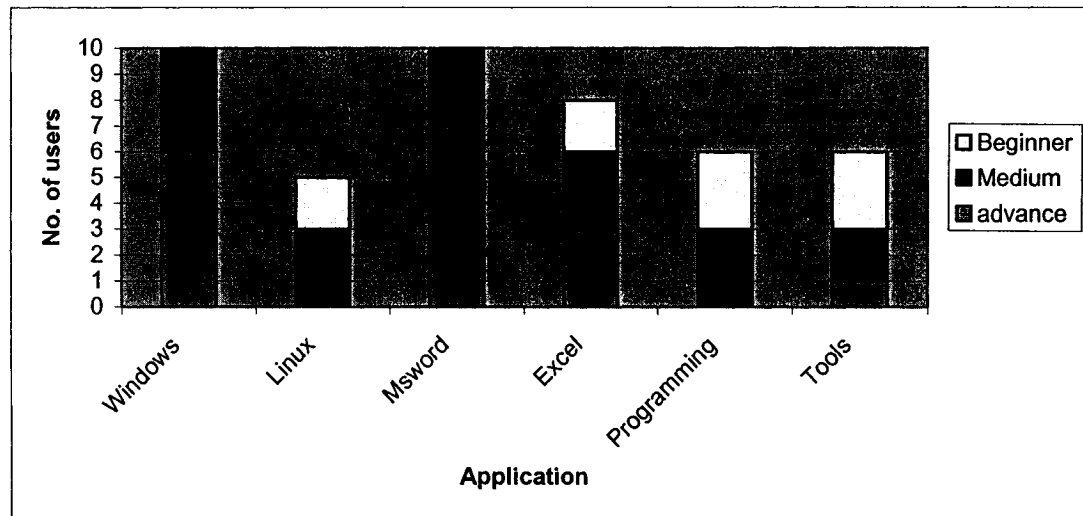


Figure5.1: Participants' computer expertise.

5.1.2. Test Results

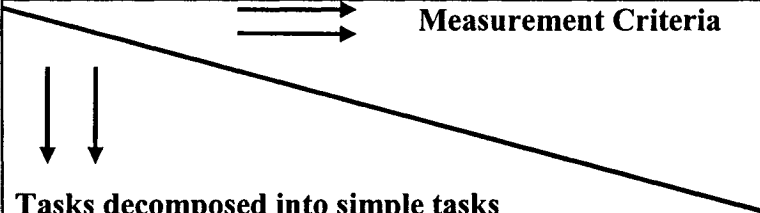
As stated in Chapter 4, when users were performing the given tasks, the interactions with the tools were captured through Camtasia [<http://www.techsmith.com/products/studio>], a screen capturing software developed by techsmith. In the previous chapter, evaluation tables for both KA tools (Protégé, PCPACK) were developed against a set of criteria for

logging the test data in a structured way. Replaying the task session activity through Camtasia player, the researcher filled out the evaluation table for every individual user. After developing one for each participant, they were combined into a single table where the designated cell entries were no longer interaction entries, but they were the number of users' who performed a particular simple task successfully. The combined tables 5.1 and 5.2 for Protégé and PCPACK KA tool are presented in the following.

	Measurement Criteria				
Tasks decomposed into simple tasks	Available users	Able to locate	Able to perform without help	Able to locate	Able to use with help
Task1: Create a project		-	-	-	-
Start the tool from the startup menu	10	10	10		
Use Save button (b)/Menu (m) option	10	6	6		
Giving file name	6	6	6		
Locate the project destination	6	3	3		
Task2: Create the root level class for the taxonomy		-	-	-	-
Select the parent Class (THING) for creating the class	10	10	7	2	2
Select the command to create the class	9	9	9		
Name the class	9	9	9		
Task 3: Create a subclass for a class in the given taxonomy		-	-	-	-
Select the parent class of the subclass	10	10	8		
Select the create command	8	8	8		
Task 4: Create multiple inheritance for a class		-	-	-	-
Select the class for Multiple inheritance	10	10	10		
Select the super class pane	10	2	2		
Select another super class for multiple inheritance	2	2	1	1	0
Task 5: Delete classes		-	-	-	-
Select the class to be deleted	10	10	10		
Select the option to delete the selected class	10	10	4	3	3
Task 6: Create a slot					
Select the class	10	10	10		
Select the button for creating template slot	10	8	8		
Name the slot	8	8	8		
Task 7: Specify the artifacts of a slot					
Select the type of the slot	10	4	3	1	1
Specify the cardinality	10	5	2	1	1
Specify maximum, minimum (if the type is integer)	NA				
Specify template value	10	5	2	1	1
Specify Default value	10	5	3	1	1
Specify inverse slot (if applicable)	NA				

Task 8: Attach a slot with a class		-	-	-	-
Select the class	10	10	10		
Use the command to add the slot at class level	10	6	3	2	2
Select the slot	5	5	5		
Task 9: Delete slots		-	-	-	-
Select the specific slot	10	10	4	4	3
Use menu option to delete the slot	7	6	6		
Delete the slot	6	6	6		
Task 10:Customize a form		-	-	-	-
Select the proper browsing key	10	5	5		
Select the form customization	10	3	3		
Select the widget	10	10	10		
Move the widgets in the suitable place	10	10	10		
Select the suitable widget type	10	7	2	3	3
Task 11: Insert an instance to the knowledge base		-	-	-	-
Select the class for which instance need to be inserted	10	8	8		
Select and press the button to insert an instance	8	8	8		
Insert the values in the corresponding slots for that class	8	8	8		
Task 12: Create a query		-	-	-	-
Select the class	10	10	10		
Select the slot in the class	10	10	10		
Select the logical condition	10	10	10		
Set the comparison value	10	10	6	2	2
Execute the query	8	8	8		

Table 5.1: Combined table for Protégé KA tool

 Tasks decomposed into simple tasks	Measurement Criteria				
	Available users	Able to locate	Able to perform without help	Able to locate help	Able to use with help
Task1: Create a project with ladder tool		-	-	-	-
Start the ladder tool	10	10	10		
Select the menu option/toolbar icon to create knowledgebase	10	10	7	3	3
Give name and password for knowledgebase	10	10	10		
Select the ontology for the knowledgebase	10	6	6		
Task 2: Create a new ladder		-	-	-	-
Locate the option/icon for creating new ladder	10	10	7	3	3
Give name of the ladder	10	10	10		
Specify the relation	10	3	3		
Task 3: Create a root for the ladder		-	-	-	-
Select the icon/ give command for creating a root.	10	10	5	2	2
Name the root object	7	7	7		
Task 4: Create children for parent node in the ladder		-	-	-	-
Select the icon/ menu command for creating child node	10	7	4	3	3
Specify the relation between parent and child node	7	2	2		
Name the new child node	7	7	7		

Task 5: Create multiple inheritance for a node		-	-	-	-
Select the second parent node	10	4	4		
Select the tangled command	4	1	1		
Select the command for creating child from second parent node	4	4	4		
Locate the child node that needs multiple inheritance	4	4	2	2	0
Select the option (from the check box) and activate the child creation	4	0	0		
Task 6: Delete a node		-	-	-	-
Select the object that has to be deleted	10	10	10		
Locate the command for deletion from Menu/icon	7	7	4	3	2
Task 7: Assign the attributes to the concept		-	-	-	-
Select the node for attaching attributes	10	7	7		
Select the attribute command from popup menu to create attribute	7	4	4	2	2
Give attribute name	6	6	6		
Task 8: Create the annotation template for a node		-	-	-	-
Select the node	10	10	10		
Activate the annotation template	10	7	4	2	2
Write the code for the template	6	6	3	3	1
Task 9: Insert instances to the knowledge base		-	-	-	-
Convert the leaf object node into instance	10	5	3	2	2
Select the command to add attribute value in the instance	7	7	5		
Task 10: Modify the knowledge base with matrix tool.			3	5	3
Select the command for creating attribute matrix	10	10	5	3	1
To add/ remove attributes in X the axis	6	6	3	2	2
To add/remove concepts in the Y-axis	6	6	6		
Add/remove attributes corresponding to the concepts in KB	6	6	6		

Table 5.2: Combined table for PCPACK KA tool.

The above combined tables [Table 5.1 & Table 5.2] present the number of users corresponding to the simple tasks. The content of these columns are described in the following:

1. First column provides the number of users who were available for a particular simple task. For a given test task, table 5.1 shows that the number of available users in the first few simple tasks are almost 10, but the number of users decreases as the simple tasks proceed because a number of users face difficulties in the first few simple tasks, therefore; they abandoned the given task. For this reason, they were not available for the remaining simple tasks of a particular task. Also, there were few tasks where

difficulty in one simple task didn't create problem for the preceding ones. Therefore, the number of available users remained unchanged in the subsequent simple tasks of a particular task.

2. The data in the second column indicates the number of users who were able to locate the particular simple task. This number can be less than or equal to the number of available users for the particular simple task.
3. Third column indicates the number of users who were able to perform the located simple task. It can be less than or equal to the number of users indicated in the second column.
4. Fourth column indicates the number of users who were able to locate the required help. The users who were unable to perform the located simple task tried to locate help from the help system.
5. Fifth column indicates the number of users who were able to perform the simple task with the located help.

From the combined tables [Table 5.1 & Table 5.2] I have computed the usability metrics: (i) functional accessibility, (ii) functional understandability, (iii) help accessibility, and (iv) help efficiency that were defined in chapter 4. The computed results of these metrics have been presented in figure 5.2. The result [Figure 5.2] shows that both tools are somewhat high in terms of function accessibility (Protégé 0.77125, PCPACK 0.6835) and function understandability (Protégé 0.835, PCPACK 0.7173). These metrics show that Protégé performed comparatively better. In terms of help accessibility (Protégé 0.5913, PCPACK 0.6627) both tools performed relatively low. In this case, PCPACK performed

comparatively better than Protégé. Such low help accessibility for Protégé is not surprising, because it is a free tool and its maintenance level is somewhat lower. Due to the lack of “search” feature in both tools many users failed to locate the required help topics.

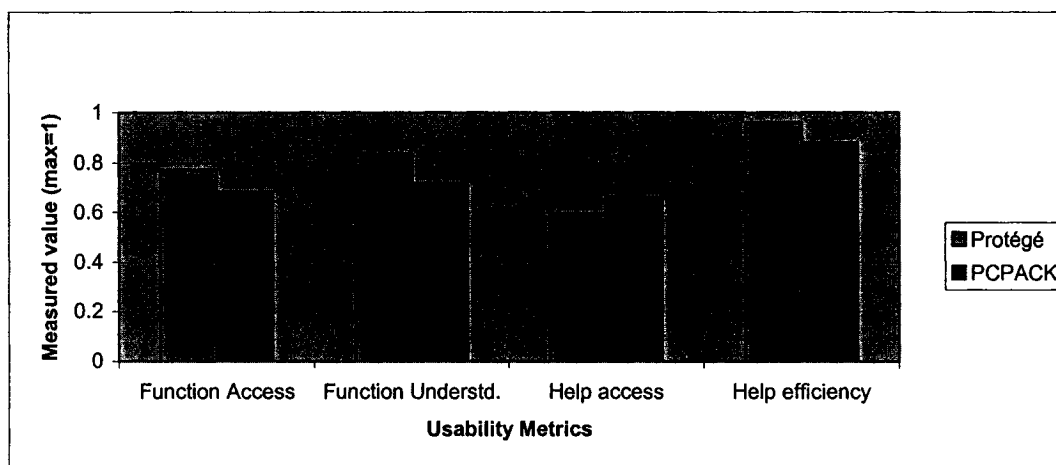


Figure 5. 2: Usability metrics for Protégé and PCPACK.

5.1.3. Satisfaction Questionnaire

Immediately following the usability session, participants were given a post-test questionnaire. Two types of questionnaires were given. First, every user was asked to answer two questions: (i) Are any parts of the interface confusing and difficult to understand? (ii) What is the worst aspect of the interface? The responses of these questions are presented in Table 5.3 and Table 5.4 respectively. The second part of the questionnaire was satisfaction ratings of different aspects of both knowledge acquisition tools, in which users were asked to rate every aspect on a scale of (1 to 7), where 7 is

easy to use, 1 is difficult, 5 is somewhat easy, and 3 is somewhat difficult. The mean satisfaction scores in different aspects of these tools are presented in Table 5.5.

Specifying artifacts for Protégé slots (3)
Confusion between knowledgebase and ontology (3)
The term ontology in both tools is confusing (6)
Many tools in PCPACK (4)
Options in PCPACK for importing and opening the knowledgebase (1)
Many options for creating PCPACK knowledgebase (2)

Table 5.3: User's comment on difficult and confusing aspects (entries in the bracket indicates the number of users commented).

Visibility of Protégé (7)
Help System of Protégé (4)
A number of panes with each tab, some of them are not initially visible (2)
Storage format of Protégé and PCPACK (3)
Writing annotation template (3)
PCPACK toolbar icons are very close to each other (5).

Table 5.4: Worst aspects according to user comment (entries in the bracket indicates the number of users commented).

Aspects of the KA tool	Protégé (mean satisfaction score.)	PCPACK (mean satisfaction score)
Starting the tool.	6.376	5.7532
Creating ontology/ladder.	3.375	4.0934
Customizing form/ annotating the tool.	6.5273	4.575
Inserting instances.	6.325	5.333

Aspects of the KA tool	Protégé (mean satisfaction score.)	PCPACK (mean satisfaction score)
Selecting icon from the toolbar.	4.666	5.025
Using Menu bar and popup menu.	6.7512	4.152
Understanding naming/ labeling.	4.33	5.251
Overall	5.735	5.336

Table 5.5: Protégé and PCPACK Satisfaction scores (In the scale of 1 to 7, 1 is difficult, 7 is easy, 3 somewhat difficult, 5 is somewhat easy)

Table 5.5 presents the mean satisfaction scores in different aspects of the Protégé and PCPACK KA tools. Overall, the satisfaction scores (Protégé 5.735 and PCPACK 5.336) for both tools were somewhat easy to use or better. Among the Protégé aspects, creating ontology (3.375) seems somewhat difficult; understanding labels (4.33) and selecting icons from the toolbar (4.66) are average; and the other aspects (starting the tool, customizing the form, inserting instances, and using the menu bar and the popup menu) are somewhat easy to use or better. On the contrary, most of the aspects of PCPACK KA tools, except creating the ladder and using the menu bar are somewhat easy to use. Creating the ladder (4.093) and usage of menu bar (4.153) seems somewhat difficult to use.

5.1.4. Observed Problems

Usability testing revealed many problems in the KA tools (Protégé and PCPACK). These problems were gathered through observing the test session, collecting user comments, and analyzing the tabulated data. The following are the top five problems for each tool

(Protégé and PCPACK), which have been collected from the observation of the test session and user comments.

Protégé

- P1.** Six out of ten participants did not notice the slide bar arrows to increase and decrease the window size, because the arrows have a low degree of color contrast with respect to background. Also, they are small in size. Therefore, it remained unnoticed to the participants.
- P2.** Participants who had no programming knowledge faced problems in understanding a number of terminologies (e.g. class and instance type in the slot definition) used in the interface. Similarly, many users also failed to understand the usage of the inverse slots.
- P3.** Search features in help systems are used to locate the required help notes on specific topic. In the help system, most of the users were looking for a search option to locate the required help topic. For example, in creating multiple inheritances, a number of users were trying to get the required help from the help system; however, the lack of search feature prevented them from locating the help notes.
- P4.** The super class pane remained hidden with the default class tab; for this reason, many participants had problems in locating the super class pane.
- P5.** Many participants commented on the poor or lack of attractive user interface. They mentioned that they failed to identify many objects in the interface due to

low visibility, and poor visual cues. They also mentioned that they could have done the tasks more easily if they were able to see the objects in the interface.

PCPACK

- Q1.** Many participants were confused about using the “Import” and “Open” option for loading and opening the knowledgebase into the ladder window. Most of the users knew the “Open” command for both loading and opening a file. For this reason, they tried to use the open command to load and open the knowledgebase. Users also thought that the purpose of the import option was to import a knowledgebase from a system other than PCPACK.
- Q2.** All participants commented that they didn’t understand the purpose of the ontology-browsing window. Some of them also added that they didn’t understand the purpose of the ontology template in creating a knowledgebase in the beginning of the project.
- Q3.** Participants expressed their concern over the icon density of the toolbar. We observed that, due to high display density, participants failed to identify the required object from the toolbar. Also, many participants expressed their concerns over many features and functionalities of PCPCAK KA tools, over which they lost some time in the system.
- Q4.** Participants had difficulties creating annotation templates for the concepts. Problems arise due to the requirement of writing template code using symbols. Users who had programming backgrounds easily understood the help notes for

creating a template, but those who were not familiar with programming concepts failed to create the template.

- Q5.** Participants had difficulties creating the relationship for the ladder. They thought the option would be available either in the toolbar or in the menu bar. In fact, such a frequently used option was not directly available; rather, it was a part of the property window, which is difficult to discover.

5.2. Analysis of Usability Problems

In the context of user interface evaluation, current research suggests that, in addition to collecting usability problems, much can be gained by analyzing the problems themselves [Keenan et al., 1999]. In this section, we follow three steps: first, classifying the problems according to common usability issues; second, interpreting the issues in accordance with the context of the problems, finally, commenting on the effects of problems on the usability principles.

5.2.1. Classification of Usability Problems

Many usability evaluation techniques, such as checklist, and heuristics are used to classify and analyze usability problems. In addition, [Susan, L. Keenan, et. al. 1999] suggested a framework to classify usability problems based on the problems. Using the framework (<http://home.townisp.com/~keenan/upt/fastpage.htm>), the problems found in the KA tools (Protégé and PCPACK) has been classified. In the following [Table 5.6], problems concerning Protégé and PCPACK are presented with the corresponding usability issues:

USABILITY ISSUE	PROTÉGÉ PROBLEMS	PCPACK PROBLEMS
1. Object appearance	<ul style="list-style-type: none"> a) "Move to upper level" and "Desktop" icon in the open/save dialog window is not consistent with similar windows icons. b) Colorblind users had problem in distinguishing system objects and created objects. c) Slide bar arrows mostly remained unnoticed. d) Notes ("c,x") icons remained unnoticed because of their low visibility, and small size. e) Default font size is relatively small in compare to other windows applications. f) Labels (Role, Type) and their values have similar foreground and background color. g) "Delete" button to delete a class/slot in the toolbar/main menu is not available. 	<ul style="list-style-type: none"> a) Annotation icon remained unnoticed. b) Participants said it would have been easy to create annotation template if there were a symbol toolbar in the annotation template tool. c) Tangled and untangled icons have poor color contrast.
2. Object layout	<ul style="list-style-type: none"> a) Mapping difficulties in specifying cardinality. b) Super class pane is not by default visible. c) Browsing key and widget types are placed in such a location that they remained unnoticed. d) User expressed their dissatisfaction in the aesthetics (i.e. color contrast and organization of labels and icons) of the user interface. 	<ul style="list-style-type: none"> a) High density of objects in the PCPACK user interface made users confused.
3. Wording	<ul style="list-style-type: none"> a) Error message needed to prevent entering data for maximum and minimum box other than integer type. 	<ul style="list-style-type: none"> a) Users failed to understand the help notes in writing the annotation template.
5. Labeling /Naming	<ul style="list-style-type: none"> a) Users confused with the usage of similar name "reference" at "Class relationship menu" and at "Class button". b) Users had wrong perception in 	<ul style="list-style-type: none"> a) Imprecise word "Analyze text" to start protocol tool from the tool launcher. b) Participants get

	<p>the label "Role".</p> <p>c) Users failed to understand the meaning of the tool tips (View selected slots) and (view selected slots at class).</p> <p>d) Users didn't understand name, such as Abstract/Concrete class.</p> <p>e) Types of the slot are not easily understandable.</p>	<p>confused in using open and import option for loading and opening knowledgebase in the tool window.</p>
6. Visual cue	<p>a) Users thought the purpose of the reference "↗" (view instances that reference a selected class) symbol is to go upper level in the class hierarchy.</p> <p>b) Users failed to recognize the purpose of "⌵" (view slot) and "⌵" (view slot at class level) icons.</p> <p>c) Confusing "+" symbol for attaching objects. ("+" generally used in creating object).</p> <p>d) Participants were not able to recognize form (⌵ (Default layout), ⌵ (layout options like)) icons for specifying the customization.</p> <p>e) Confusing cue "+" instead of "browse" in locating the destination of the project during the project save.</p>	<p>a) Users failed to identify the root node icons from the toolbar.</p> <p>b) Many users selected the "Move up" arrow for creating a node instead of using the "ladder up" from the toolbar.</p> <p>c) Users failed to identify icons "Ladder up", "ladder down", "Tangled ladder", "untangled" from the toolbar.</p>
7. Cognitive aspects	<p>a) To create root level classes for taxonomy, selecting a system class as parent was not well perceived with the user's cognitive task.</p> <p>b) In multiple inheritance, Locating second parent for a class don't match with user's mental model (user thought some drag type relationship).</p>	<p>a) Users expected a kind of tabular structure for entering instances.</p> <p>b) Creating multiple inheritance didn't not match with user's cognitive model (Users expected drag and drop type of connection).</p> <p>c) Users could not understand how to create multiple roots for a ladder.</p>

8. Miscellaneous.	a) No “search” option in the help system.	<ul style="list-style-type: none"> a) No “Search” option in the help system. b) No “Undo” option to reverse the knowledgebase. c) No feedback when original knowledgebase modified and lost the original knowledge entries. d) Participants didn’t understand the purpose of ontology browsing window and ontology template. e) Locating annotation template for a node f) Problems in locating the option for converting leaf object into instance.
-------------------	---	--

Table 5.6: Usability issues for Protégé and PCPACK KA tools.

5.2.2. Interpretation of Usability Issues

Table 5.6 presents usability problems with the corresponding usability issues [Keenan et al., 1999] for both KA tools. Although the frequency of problems for an issue may be lower for a particular tool, caution must be used in judging one tool as more “usable” than the other based on the number of problems with a specific issue because (i) users may have had a number of different usability problems in different contexts, (ii) the severity of the problems is important in comparing tools, and (iii) some usability issues may be strictly user-related, thus inflating the number of issues for software.

1) Object Appearance

Participants had many object appearance problems in Protégé KA tools, some of which, namely “inconsistency” with the windows icons, and missing the “delete” option in the toolbar and menu bar were repeatedly faced by many users. Figure 5.3 shows that icon: (i) going one level up, (ii) create new folder, and (iii) desktop, are not consistent with the *de facto* standard (windows icons). For this reason, many participants had difficulty identifying such icons. However, consistency in object appearance is equally important for both expert and novice users. Inconsistency leads expert users to face difficulties in performing the task efficiently, and novice users to face difficulties in learning a tool. Since the majority of Protégé users are intermittent users and have a low degree of computer expertise, difficulty in learning the tool is a major concern for the Protégé KA tool.

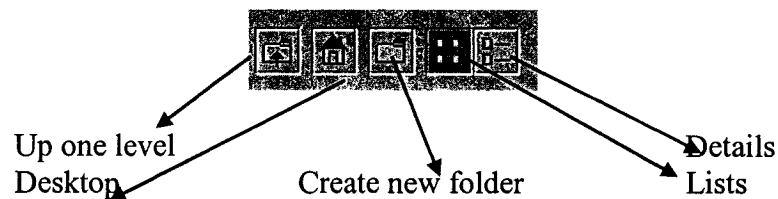


Figure 5.3: Protégé metaphors in save dialogue.

It was also found that many users were looking for the “delete” option in the toolbar and in the menu bar for deleting a class or a slot object, but this option was available only with the popup menu. Comparatively, more users had difficulties finding the “delete” option for the slot than for the class. In deleting a slot, users first tried to use the “delete” button from the toolbar, after which, they tried to delete the slot from class level from where they created it. In fact, slot must be selected from the slot list within the slot tab.

Perhaps, by considering the low frequency of the “delete” option compared to the “create” option, or by considering the consequences of an unexpected delete action, the designer chose not to place this button in front of the interface. Since most systems keep the “create” and “delete” option together, maintaining consistency with other systems is important for making the interface easy to use and easy to learn.

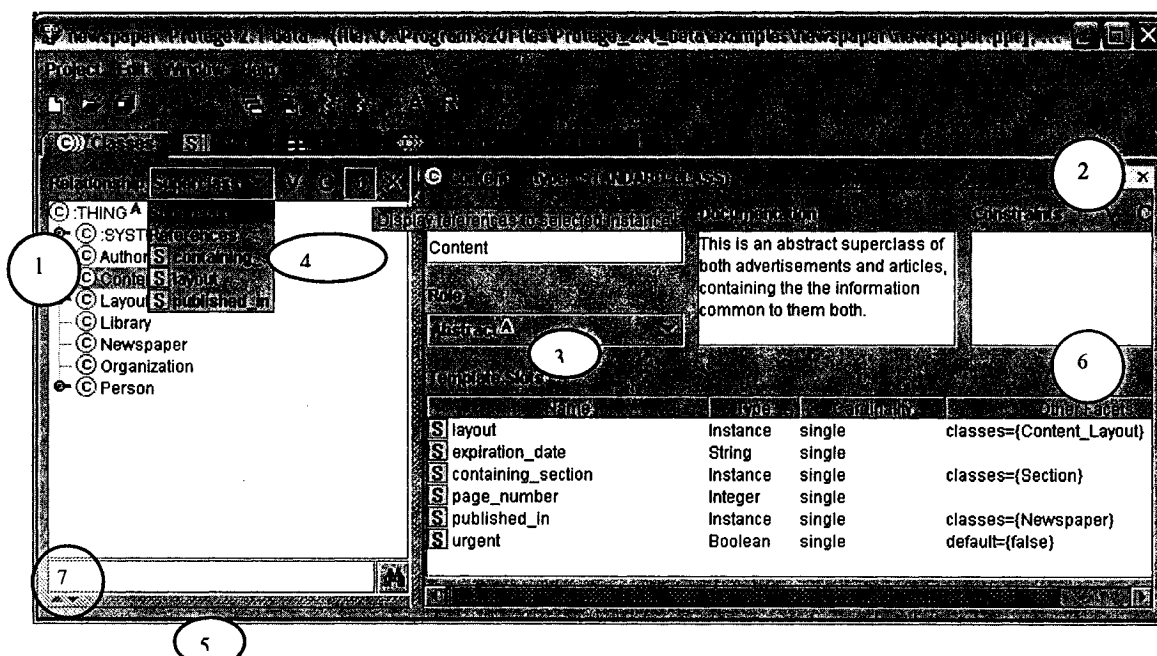


Figure 5.4: Protégé user interface.

Many other object appearance problems are as follows: (i) the “notes” icon in the top-right corner remained unnoticed, (ii) drop-down lists, such as “Role” could not draw the users’ attention, and (iii) colorblind users couldn’t distinguish between system objects and user created objects. These problems are mostly attributable to the low degree of color contrast, and to the small size. For example, a drop-down list, such as “Role” could not gain users’ attention as it has the same color for both the drop-down list item and for its label. Also, Protégé tools uses colors (yellow and pale) to distinguish between system

objects and user created objects. The low contrast between the colors (indicated in circle 1 [figure 5.4]) prevented partially colorblind users from being able to distinguish between them.

PCPACK users faced many similar object appearance problems due to color contrast and to size. Among them, one of the frequent PCPACK problems was identifying tangled and untangled icons. There might be several reasons behind the failure to identify tangled and untangled icons, but most users mentioned that the low degree of color contrast block their visibility. Although the pop-up menu provides the option to create an annotation template, many users were looking for a template creation icon in the toolbar. Although there was an icon in the interface, most users didn't find that template icon.

The majority of the object appearance problems in both Protégé and PCPACK were due to color contrast and to the size of the objects. Once user locates and overcomes the above-mentioned problems, s/he can easily use the function second time. As a result, regular users of these tools can easily overcome these problems (except that of colorblindness); however, most of the KA tools' users are intermittent rather than regular, so these problems pose setbacks in using these tools. In this regard, the proper use of color in the user interface object can facilitate the users' identification of and usage of these objects. Conversely, if used inappropriately, color can be distracting and can disrupt the visual search.

2) Screen layout

Screen layout refers to how the user-interface objects are laid out on the screen. A consistent layout helps users to predict where to find information and where the particular object is located on the screen. Generally, objects are placed on the screen based on different guidelines [Nielsen, J. 1994], but there is no agreement among people. For this reason, it is very difficult to place objects in an order that will satisfy all users. In this way, the screen layout for KA tools is a challenge because it involves users from different backgrounds.

After the test, when Protégé users were asked why they didn't use the objects such as (i) browsing key in the form screen, (ii) super class pane with the class tab, and (iii) widget types in the form, they stated that these objects were placed in a way that made locating them and using them difficult. These objects were not placed in the proper location, and, they had a low degree of color contrast; therefore, they failed to grab the users' attention. Moreover, all the participants except one found specifying the "multiplicity" of the objects problematic because of the difficult visual mapping of the related options in the interface. Users expressed dissatisfaction with Protégé's aesthetics. Such negative impressions about a system discourage users from using the system in future. On the other hand, PCPACK users reported high display density, the amount of information on a display and how tightly packed it is in relation to the size of the display [Jones, M.K., 1989], created problem in selecting the objects from the interface. The problems mentioned above have little impact on the users who have a high degree of computer

literacy because they can overcome most of the problems; however, users who have minimal computer expertise can suffer in learning and using the tools.

The screen consists of different types of resources, such as icons, buttons, the toolbox, and text. Usage of these components revealed that PCPACK had very few problems compared to Protégé. This is not surprising, as Protégé is a free tool managed by a research group, whereas, PCPACK is a commercial tool in which high quality visual interface is important for attracting users.

3) Labeling/Naming

Telling people what will happen before they click on a new object is a basic, long-established principle of user interface design [Nielsen, 1994]. But, until now, there has been discrepancy between the designer's interpretation and user's interpretation of the labels that are common in most software systems. Most of the time, such discrepancies create major problems for end-users. In the context of KA tools, of which the majority of users are end-users, the labeling of objects is an important usability issue.

Two major types of labeling problems, (i) confusing similar names for two different objects, and (ii) having a non user-centered name for the problem domain, were observed in the Protégé and PCPACK interfaces. Most participants were confused about using two labels in Protégé. One is a "reference button" in the class button to display the instances that refer to the selected objects, and the other is the "reference" option in the class pane that indicates the reference classes for the selected class. Most users selected one instead

of the other. In the same way, PCPACK users were confused about the “import” and “open” options, which have been used for loading and opening any knowledgebase. To many users, “open” mostly meant loading and opening a file within the system, and “import” was for importing a file/project from another system. For this reason, most users had difficulties choosing the option for opening a project. This type of confusing labels created problems for individual users to learn the system. This is also problematic for intermittent users of the tool.

Due to the lack of user-centered naming, users had problems with understanding the labels. For example, many users thought that the purpose of the label “Role” was to describe the purpose of the particular class, whereas the actual purpose was to define whether a class was of the abstract or concrete type. Also, they failed to understand the meaning of the tool tip, “{View selected slots}” and “{view selected slots at class}” and terminologies such as “Abstract/Concrete” class and slot types.

For any system, user-centered naming is important in order for end-users to understand and use the system easily. KA tools are kind of systems that borrow many terminologies from the knowledge engineering discipline in order to identify the objects and the system functionalities. From this perspective, user-centered naming is more important for end-users of KA tools because most users are not familiar with knowledge engineering terminologies. Designing appropriate user-centered labels can provide easier user interaction with the system, which in turn will increase the efficiency, and the learnability of the system.

4) Visual Cue

Visual cues are provided in the user interface to show the user the existence of individual user-interface elements and the operations that those objects afford [Jones, M.K., 1989]. Even though most developers use visual cues in the interface objects, using the appropriate visual cue that will become part of the user's cognitive model of the system is one of the most challenging tasks [Jones, M.K., 1989].

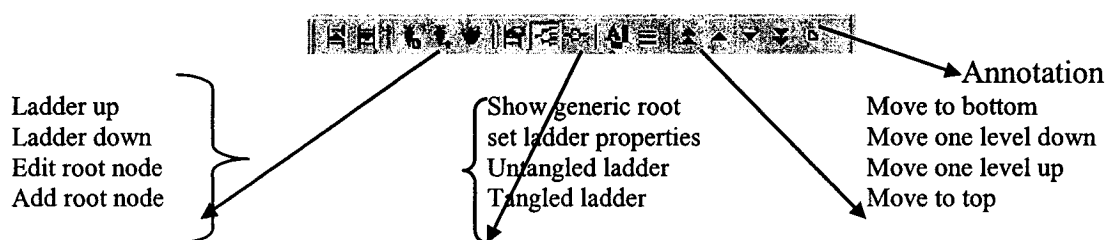





Figure 5.5: PCPACK ladder toolbar icons.

Many different visual cue problems have been recorded in Protégé. Among the observed problems, the back-reference button provided a misleading cue. All of the test participants first thought that the purpose of the reference symbol (display objects that reference the selected class “”) is to go to an upper level in the class hierarchy, as a similar symbol was also used in other places of Protégé to go “one level up” in the directory hierarchy; therefore, users get confused when using this symbol. Also, the symbols, ‘+’ and ‘C’ have been used with most features for adding/attaching and for creating an object respectively. A number of users were confused with the usage of create vs. add symbol because they were familiar with ‘+’ for creating an object rather than for adding one. In addition, problem also arises when the ‘+’ sign is used to locate a folder instead of browse. In customizing the screen for a particular class, none of the ten users

used the form layout ( (Default layout), (layout options like)) icons for specifying the customization. When they were asked why they didn't use these, they stated that they didn't understand the purpose of these icons based on their appearance.

In PCPACK, although many icons have been used to represent real life concepts, users frequently made mistakes in selecting and using these icons. For example, icons used for creating and deleting a child node, for creating a ladder, and for creating a root node didn't match well with the users' cognitive model, knowledge that the user gains through the experience of using other systems. We have often observed many times users trying to use one icon instead of another. For example, in creating a child node, instead of using the "ladder up" icon most users tried to use the "move up" icon, which is for moving one step in the ladder. Also, all users had difficulties identifying the root node icon on the toolbar.

The above description shows that improper use of visual cues in both PCPACK and Protégé created many problems, a number of which created difficulties in the initial learning stage.

5) Cognitive Aspects

In addition to visual cue problems, users of PCPACK and Protégé had many cognitive problems with the interfaces. Actually, the problems were due to poor or conflicting cognitive models, knowledge that the user gains through the experience of using other systems. For example, in PCPACK, the users expected a kind of tabular structure related

to the concept tree for entering the instances that they gain from database concepts. Also, participants did not understand the purpose of the ontology browsing window and the ontology template in creating the knowledge base. In creating “multiple inheritances” using Protégé, users were trying to make a “drag-drop” relationship in order to create multiple parents for a class, but the relationship needed to be created through the super class pane, which was invisible in the class tab. In creating multiple inheritances, most users failed to understand the necessity of the super class pane and its usages.

The Cognitive aspects of any interface are important, because a sequence of task activities in a system conform to the user’s cognitive model of the task makes the system easy to use. On the other hand, the conflicting model makes the systems difficult to learn. This sort of problem is more difficult for novice users because these problems contribute to learning difficulty and to a reduction in efficiency for users who have experience with other KA tools.

6) Miscellaneous

A number of problems were found that do not fit into any of the above categories, especially for PCPACK users. The users of both tools felt the necessity of the search feature to look for help topics in the help system. The other problem that creates a major drawback in PCPACK system is the absence of an “undo” option, without which, participants failed to retrieve lost knowledge entries from the knowledgebase. Also, the automatic updating in the knowledgebase without any system feedback is another major concern for PCPACK users.

5.2.3. Usability Principles

Required usability attributes, including learnability, understandability, and satisfaction have been described in Section 3.2. Usability assessment scores and most of the common problems analyzed in the previous sub-section will be used to understand the current state of usability of these tools.

Understandability: Understandability is a measure that indicates how well a new user will be able to understand whether the software is suitable and how it can be used for particular tasks (ISO 9126-2). Protégé and PCPACK tools have somewhat better understandability to locate the functions (Protégé 0.77125, PCPACK 0.6835) and to use these functions (Protégé 0.835, PCPACK 0.717). The interpretation of results reveals that their poor cognitive aspects and visual cues make PCPACK tools comparatively difficult to understand. Since the users are not regular users of this tool, it is likely that they will face cognitive problems each time when they return to this tool.

On the contrary, Protégé suffers from visual cues and labelling problems that most users were able to overcome resulting in better performances. Non-user centred labelling, particularly knowledge engineering terminologies in Protégé, created challenges in understanding the system functionalities and documentation.

Learnability: Learnability is a measure of how quickly a user can become proficient at using the system (Nielsen, 1993). Since participants in this study were not experienced in these tools, the learnability of these systems was a major issue. Learnability was measured in terms of help accessibility and help efficiency. In terms of help accessibility

(Protégé 0.5913, PCPACK 0.6627) both tools performed relatively low. But in terms of help efficiency (Protégé 0.96 and PCPACK 0.88) both tools performed better. Learnability is strongly related to understandability, and the measures of understandability can be indicators of learnability (ISO 9126-2, 2002). The understandability metrics indicate the level of ease with which one can learn these tools.

The analysis of problems indicates that the visual cues of the interface objects and cognitive problems created a major challenge for beginners who were attempting to learn the KA tool PCPACK. Also, the analysis showed that some of these problems had high severity, which hindered the normal task flow in most cases.

The learnability of the inconsistent icons and the non user-centered labeling in the Protégé interface, however, were issues for some participants. The usage of inconsistent icons in the interface created a major challenge for participants who had experience with Microsoft products. The non-user-centered labeling and wording created setbacks for end-users. The difficulty in learning was increased when labeling and wording are related to Knowledge engineering terminologies.

Satisfaction: The results of the post-test questionnaire examined the participants' overall experience with the Protégé and PCPACK KA tools. Overall, satisfaction ratings from the post-test questionnaire suggest that these tools are somewhat easy to use with Protégé scoring 5.735, and PCPACK, 5.336. Also, the participants reacted positively to most of the aspects of both tools. Creating ontology/ladders, using PCPACK menus, and

navigating toolbars seem the most difficult in users' satisfaction ratings. During the satisfaction questionnaire, most users expressed their dissatisfaction with Protégé aesthetics and with PCPACK's abundant functions and features.

5.3. Lessons Learned

The process of designing usability tests, and collecting and analyzing data provided the authors with important experience in usability testing. From the testing, we learned that the selection of critical tasks and representative users in accordance with the test criteria is one of the important pre-conditions for gathering useful data to provide recommendations for further improvements to the system.

Identifying problems that the test users were having with KA tools (Protégé and PCPACK), and exploring the issues arising from the problems was complex because three different categories of issues that provided us with valuable lessons related to user study and to KA tools arose from this investigation. First, problems reported in the previous sections were mostly related to the specific tool.

Second, we identified typical user issues that come with most usability studies.

- Users brought their biases and experiences with them
- Users compared the given interface to the interfaces that they have used before, and these comparisons revealed a great deal about an interface's usability
- Users if motivated to use the tools could have tried more features and functionalities, and thereby more problems could have been reported

Finally, we discovered few issues that were purely related to the development of knowledgebases using both tools.

- The sequence of task activities in the system mostly differed with the users' cognitive model of the task.
- Limited familiarity with the knowledge engineering concepts leads to the misunderstanding of naming and wording that were used in KA tools' interface.
- Users required a detailed understanding of the knowledge model (ontology/ladder) before creating a knowledgebase structure, because most participants had difficulties in understanding and using the knowledge model.

5.4. Recommendations

The usability study of Protégé and PCPACK KA tools (described in chapter 4) produced a set of usability metrics along with a set of usability problems, which were presented in section 5.1. The metrics can be used as benchmarks for further study of KA tools. To alleviate the observed problems, we conducted an analysis of these problems in section 5.2. The analysis of these problems revealed two different types of problems: (i) that are common in both tools, and (ii) that are explicitly related to one of the particular KA tool. Based on this we make two different types of recommendations for further improvement of these tools. First, considering the commonality of the problems we make the following two general recommendations.

- 1) Test participants need to perceive and use a number of functionalities (Table 4.3 for Protégé and table 4.4 for PCPACK) to perform a KA task (Table 4.2). However, I

observed many users failed to perceive the sequence of required functionalities to perform the given tasks. Such type of recurring problems was observed in both Protégé and PCPACK KA tools. To alleviate these problems, the sequence of functional activities of these KA tools can be examined thoroughly to discover the interaction difficulties.

- 2) Analysis of results in section 5.2 indicates that participants had difficulties in understanding non user-centered labeling and wording borrowed from knowledge engineering terminologies. To alleviate these problems, the usage of user-centered labeling and wording and some background on KA process are recommended.

Second, a set of product-specific recommendations were produced based on the analysis of the results presented in section 5.2. The following actions are recommended to improve the Protégé knowledge acquisition tool.

- 1.1. *Search* option in the system help is very common and useful feature to learn and understand a particular system. However, it has been found that there is no *search* option in the Protégé help system. The help system should provide search functionality to facilitate searching by topics.
- 1.2. As most of the users are familiar with the Windows platform, metaphors used in Protégé should be consistent with those in Windows. For example, in the Save dialogue, the metaphor “Move one level up” and “create new folder” should be consistent with the *de facto* windows standard. The usage of the *de facto standard* increases the efficiency in use for both expert and end users.

- 1.3. The usage of ambiguous tool tips can create difficulties in learning and understanding the new user interface features. The analysis of results (section 5.2) indicates that a number of users were confused by ambiguous tool tips. Tool tips text messages used with icons should be precise, specific, and provide a direct reference to the icon function.
- 1.4. Drop-down lists, such as “Role,” “Relationship,” “Browse,” and “Selected widget type” remained unnoticed to the users because both the label and the drop-down list item were in the same color. For grabbing users’ attention, the drop-down item should be a different color than that of label.
- 1.5. Colors (Pale, Yellow) have been used to distinguish between systems object and user-created objects, but a number of partially colorblind users failed to make the distinction. Alternative cues should be used so that this minority percentage of users can distinguish between the objects.
- 1.6. Participants were confused using ‘+’ (to attach a slot with a class) and ‘C’ (Create a slot, class) buttons because a number of users were already familiar with the ‘+’ sign for creating objects in other applications. The use of ‘A’ (to attach an object) instead of ‘+’ will be easier for the user.
- 1.7. Many participants failed to identify “Add/ delete note” icon from the toolbar. Using a bright color for the “Add/delete note” icon and placing it in the left of the class form can focus the users’ attention.
- 1.8. Most users faced problems in specifying the cardinality of the slot; therefore, the cardinality of the slot should be re-examined.

Similarly, based on the findings from the analysis of usability results presented in section 5.2, we recommend the following improvements for the PCPACK KA tool.

- 1.1. Users frequently selected the wrong icon; therefore, further research is required to determine the icon for a specific function, so that the icon's functionality in the system is easy to recognize, and interpret, and so that the user will not have to remember complex metaphors.
- 1.2. Due to similarity in appearance and meaning most users tried to use "*Ladder up*" icon (to delete a child) instead of "*Move up*" (moving one step up in the ladder). For better understandability, it is recommended to avoid using similar appearances for different types of icons.
- 1.3. The analysis of results indicated that objects in the user failed to grab user attention due to high display density and poor color contrast. Increasing the object size, and the spacing between objects; and implementing a high degree of color contrast can make the interface objects more visible to the users.
- 1.4. To prevent the unexpected modification of knowledge bases, the system should provide feedback information during the updating of knowledgebases.
- 1.5. Having "Import" and "Open" options for loading and opening a knowledgebase confused many participants as "open" is known to be used for both purposes, and "import" is generally used for importing a project/file from another system. In this regard, it is recommended that the "open" option be used for both opening and importing knowledgebases.
- 1.6. PCPACK is used for inserting, deleting and editing knowledge entries in the Knowledgebases. For the lack of "*Undo*" option users frequently lost

knowledge entries. The addition of an “*Undo*” function in the PCPACK KA tool will allow for the retrieval of lost knowledge entries.

Although these recommendations are product-specific, and based on the results of a systematic study conducted in this thesis, designers should not view these as a comprehensive list or final solution to prevent usability issues because these are based on a segment of users (only end-users) who performed a set of test tasks in the education domain. In implementing these recommendations the designers need to be cautious so that the design change will not create any further usability problems for the users that were not considered in this thesis.

5.5. Discussions of the Study

The experimental study conducted in this research measured a number of usability attributes² and identified a set of usability problems. The findings of this study were limited in term of certain issues³. In the following, the limitations of the findings and if possible, the ways to overcome these limitations were discussed:

First of all, the purpose of this study was to investigate how end-users perform with Protégé and PCPACK KA tools. Therefore, the study used only the tasks that were pertaining to the end-users of Protégé and PCPACK KA tools. Knowledge engineers are the other types of users who perform different types of tasks. As a result, they could face

² Measures: learnability, understandability and satisfaction

³ Selection of users, available resources and experimental setup

different types of challenges that were not included in this study may affect the generalization of the current findings.

Another important user issue that limits the findings of the study is the differences of user skills. During the review (section 4.3.5) the researcher found that a majority of the participants⁴ who had positive attitudes and interest toward new computer system performed better than others⁵. Such an inclination toward a new system by the majority of the test participants decreases the reliability of the results because the real end-users rather have mixed attitudes. The larger number of users from broad user categories in terms of ages, professions and skills could provide more reliable results. The other advantage is that it would increase the likelihood of uncovering problems unique to specific types of users.

The tests of Protégé and PCPACK KA tools were conducted in two consecutive⁶ test sessions. In the second session, users became tired and less interested toward the test tasks. Therefore, instead of trying to overcome any difficulty they simply abandoned the test task resulting lower performances with PCPACK KA tool. Since users frequently abandoned tasks in the second session, a large portion of the PCPACK tasks remained inaccessible; therefore, fewer problems were noticed from their task activities in compare to Protégé. The above problems affect the reliability of the findings and can be overcome by scheduling the test of Protégé and PCPACK KA tools in two different days. If it's not

⁴ Electrical and Mechanical Engineering graduate students, and Computer Science first year Undergraduate students

⁵ Mathematics and Statistics graduate students

⁶ Protégé in first session and PCPACK in the second session

possible within the available budget, the users can be divided in two different groups. In the first session, instead of assigning all users to perform tasks with Protégé KA tool two groups can be assigned to perform tasks with two different KA tools. In the second session, they (two groups) can be assigned to perform tasks with alternate KA tools that may minimize the affects of the above-mentioned problems.

The usability attributes (subjective and objective) were measured within the scope of the available resources. Objective attributes: understandability and learnability were measured in terms of a limited number of metrics⁷ rather than all the available metrics; and, subjective satisfaction ratings were collected through a set of questionnaire. Users rated the aspects (included in the questionnaire) based only on their limited experiences with the test tasks rather than based on the experience with all interface features and functionalities. These limitations in measuring the required attributes may affect the usage of these measures as a benchmark for further usability study.

⁷ (i) Understandability metrics: function understandability, function accessibility; (ii) learnability metrics: help accessibility and help efficiency

6. Conclusions

6.1 Contributions

In this thesis we examined two KA tools for usability evaluation. We reviewed major usability techniques and methods for selecting appropriate techniques and methods for our study. For usability studies, Hix's "Checklist based methodology" has been adopted and modified for the evaluation of Protégé and PCPACK KA tools. Following this methodology, usability experiments have been designed and conducted for assessing usability and for identifying the usability problems of the two KA tools.

The usability of the two KA tools Protégé and PCPACK was studied using a population of graduate and undergraduate students for the domain of student information system. Objective and measurable usability attributes such as, accessibility, function understandability, help accessibility, and help effectiveness were measured to obtain the understandability and learnability of these KA tools. A satisfaction questionnaire was used to collect users' attitude in different aspects of these tools. In term of these attributes, Protégé seems somewhat better of the two KA tools except in its help feature. Overall, participants reacted positively to both KA tools. The study also reported in chapter 5 the critical usability problems in both KA tools. The critical problems were analyzed to determine where usability principles were violated. Product-specific recommendations were made in that chapter to help designers adhere to these principles in future improvements of these two tools. The use of product-specific recommendations in the future design can help to increase the usability of the KA tools.

The major finding from this study is that KA tools need a balance of simplicity and functionality. Although Protégé has more problems with respect to “Visualness” and language of expression, its simplicity allowed the participants to complete more tasks using Protégé when compared to PCPACK. The other important result is there is striking similarity in the identified shortcomings by the end-users’ when using these two tools.

6.2. Future Work

This study was conducted using a specific case study of student information systems where a sample set of tasks were tested with ten users. The results of this study provided specific information on the usability of knowledge acquisition tools. Although this information is helpful, it also provides opportunities for further study of the usability of knowledge acquisition tools. The opportunities include varying the population of end users to eliminate any bias; testing the features of the tools more thoroughly such as testing menu navigation, testing the usability of labels, icons and icon metaphors. Also, critical usability issues from this research can be re-investigated to understand their impact on the KA tools’ user interface.

The results of the tests indicate that participants face difficulties in selecting the sequence of task activities, understanding the labels and recognizing the icons. Further work can be done in designing user-centered tasks, icons, and labels for KA tools. It also remains to be tested the effectiveness of the product specific recommendations made in this thesis.

Reference

1. [Avouris 2002] Avouris, N.M. An Introduction to Software Usability: A report of Network of Excellence on Software Usability funded by the Greek Secretariat for Research and Technology (GSRT) 2002.
2. [Bias 1991] Bias, R. Walkthroughs: Efficient collaborative testing. *IEEE Software* 8,5 (September), 94-95.
3. [Brian 1996] Brian R. Gaines and Mildred L. G. Shaw "Eliciting Knowledge and transferring it Effectively to a Knowledge-Based System", a research report of Knowledge Science Institute University of Calgary (<http://ksi.cpsc.ucalgary.ca/articles/KBS/KSS0/>).
4. [Brian 1998] Brian R. Gaines and Mildred L. G. Shaw "Web Grid: Knowledge Modeling and Inference through the World Wide Web" A report of Knowledge Science Institute, University of Calgary (<http://repgrid.com/reports/KBS/KMD/>)
5. [Blythe 2001] Blythe, J., Kim, J., Ramachandran, S. and Gil, Y. " An Integrated Environment for Knowledge Acquisition" *International Conference on Intelligent User Interfaces*, 2001.
6. [Birmingham 1989] Birmingham, W. and Klinker, G. " Building Knowledge Acquisition Tools" A publication of Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, 1989.
7. [Butler 1996] Butler, K.A. "Usability Engineering Turns 10" *Interaction*, January 1996.
8. [Chaudhry 1998] Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P. OKBC: A programmatic foundation for knowledge base interoperability. In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin, July 1998.
9. [Cupit 1999] Cupit, J., Shadbolt, N., Cheng, P. Peebles, D. "Compiling Ontologies into Structured Views and Interviews : The Design of A Diagrammatic Knowledge AcquisitionTool" KAW99, Voyager Inn, Banff, Alberta, Canada.
10. [Carol 2002] Carol M. Barnum "Usability Testing and Research" A Longman Publications-2002.

11. [Corde 2001] Corde, R.E. Task selection-bias: A Case for User-Defined Tasks, *International Journal of Human-Computer Interaction* 13(4) 411-419.
12. [Kirakowski 1993] Kirakowski, J. and Corbett, M. SUMI: The Software Usability Measurement Inventory, *British Journal of Educational Technology*, 24(3), 210-212.
13. [Davis, 1979] Davis, R. Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence*, 12, 121--157
14. [Dix 1998] Dix, A, Finlay, J, Abowd, G, and Beale, R. "Human-Computer Interaction- 2nd Edition: Evaluation Techniques" Prentice Hall-1998
15. [Drs. 1998] Drs. Erik P.W.M. and van veenendaal CISA "Questionnaire based usability testing "in Conference Proceeding of European software quality week, Brussels, Nov-1998.
16. [Dumas 1993] Dumas, J. S. & Radish, J.A. *Practical Guide to Usability Testing*, Ablex Publishing, Norwood, NJ.
17. [Embrey 2000] Embrey, D. *Task Analysis Techniques*, A technical report of Human Reliability Associates Ltd. 2000.
18. [Folmer 2004] Folmer, and E. & Bosch, J. " Architecting for usability; a survey" *Journal of systems and software*. issue 70-1, 2004. pp. 61-78.
19. [Gennari 2002] Gennari, J., Musen, M.A., Ferguson, R.W., Grosso, W.E., Noy, N.F. and W. Tu, S.W. *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*. 2002
20. Girgensohn, A. and Shipman, F.M. *Supporting Knowledge Acquisition by End Users: Tools and Representations*, 1992 ACM 0-89791.
21. [Guillemette, R.A. 1995] Guillemette, R. A, " The evaluation of usability in interactive information systems" Chapter 13, *Human Factors in Information Systems: Emerging theoretical bases*, Ablex Publishing Corporation, New Jersey, 1995.
22. [Hayes-Roth 1963] Hayes-Roth, F., Waterman, D.A. & Lenat, D.B., Eds. (1983). *Building Expert Systems*. Reading, Massachusetts: Addison-Wesley
23. [Hix 1991] Hix, D., and Schulman, R.S. *Human-computer interface development tool: Communication of the ACM*, March 1991/Vol. 34 No. 3

24. [Hix 1993] Hix, D. & Hartson, H.R. Developing user interfaces: Ensuring usability through product and process, Chap 2. Willy & Sons, NY
25. [ISO 9241-11 1998] ISO 9241-11 Ergonomic requirements for office work with visual display terminals (VDTs) – Part II: guidance on usability (1998)
26. [ISO 9126-2] ISO/IEC 9126-2: Software engineering –Product quality – Part 2: External metrics.
27. [Jackson, 1999] Jackson, Peter. Chapter 10, Introduction to Expert Systems. Addison-Wesley, 1999.
28. [Jones 1989] Jones, Mark. K. Human-Computer Interaction: a design guide Educational Technology Publications, New Jersey.
29. [Jtirgen 1994] Jtirgen, K.B., John, M. C., Mary, B. R. and Mark K. S. "Comparative Usability Evaluation: Critical incidents and critical threads" CHI-94 [21]
30. [Jacobsen 1999] Jacobsen, N.E. " Usability Evaluation Methods: The Reliability and Usages of Cognitive Walkthrough and Usability Test" Department of Psychology, Copenhagen University, Denmark, 1999.
31. [Kline 2002] Kline, R., Seffah, A., Javahery, H., Donayee, M., & Rilling, J. (2002, September). Quantifying developer experiences via heuristic and psychometric evaluation. *Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments (HCC 2002)*, Arlington, VA, pp. 34-36.
32. [Kim, 2000] Kim, J. & Gil Y. User studies of an interdependency-based interface for acquiring problem-solving knowledge. Conference on Intelligent User Interfaces (IUI-2000), New Orleans, Louisiana.
33. [Kantner 1994] Kantner, L. " Techniques for Managing a Usability Test" IEEE Transactions on Professional Communication, Volume 37, Number 3, September 1994.
34. [Kartz] Katz B. "From Sentence Processing to Information Access on the World Wide Web" An Overview of the START System
35. [Keenan 1999] Keenan, S. L., Hartson, H.R, Kafura, D.G, and Schulman, R.S "The Usability Problem Taxonomy: A Framework for Classification and Analysis", *Empirical Software Engineering*, 4, 71-104(1999)

36. [Kahn 1984] Kahn G., and McDermott J. MORE: an intelligent knowledge Acquisition tool. In Proc. 9th International joint conference on Artificial Intelligence, pp.581-4.
37. [Kelly 1995] Kelly, T. & Allender, L. Why choose? A process approach to usability testing. In Yuichiro Anzai, Katsuhiko Ogawa, & Hirohiko Mori (Eds.), Symbiosis of human and artifact: Human and social aspects of human-computer interaction proceedings of the sixth international conference on human computer interaction, Tokyo, Japan, 9-14 July 1995, Volume 2 (pp.393-398). Amsterdam: Elsevier.
38. [Lansdale 1994] Lansdale, Mark. W., and Ormerod, Thomas. C. "Understanding interfaces: A Handbook of Computer Dialogue", Academic Press, 1994.
39. [Lea, 1988] Lea, M. Evaluating user interface designs. In T. Rubin (Ed.), user interface design for computer systems (pp. 134-167). Chichester, England: Ellis Horwood.
40. [Lewis 1982] Lewis, C. Using the "Thinking Aloud" method in cognitive interface design. Research report RC9265, IBM T.J. Watson Research Center, Yorktown, NY.
41. [Matera 2002] Matera, M., Costabile, F. M., Garzotto, F. and Paolini, P. SUE Inspection: An effective method for systematic usability evaluation of Hypermedia, IEEE transactions on systems, man, and cybernetics- Part A: System and humans Vol. 32 No. 1, January 2002.
42. [Mayhew 1999] Mayhew J. Deborah "The Usability Engineering Life Cycle: Usability Goal Setting" Morgan Kaufmanns Publishers, INC- 1999.
43. [McGraw 1989] McGraw, L.K. Harbison-briggs, K. "Knowledge Acquisition: Principles and Guidelines" Prentice Hall, Englewood Cliffs, New Jersey, 1989.
44. [Mildred L.G] Mildred L. G. Shaw and Brian R Gaines "WebGrid: Knowledge Elicitation and Modeling on the Web" (<http://ksi.cpsc.ucalgary.ca/articles/WN96/WN96WG/WN96WG.html>)
45. [Maulsby, S., 1993] Maulsby, S., Greenberg, S., & Mander, R. Prototyping an intelligent agent through wizard of oz. In INTERCHI-93

46. [Mack 1994] Mack, R. L. & Nielsen, J. (1994). *Usability inspection methods*. New York, NY: John Wiley & Sons. ISBN 0-471-01877-5.
47. [Meij 1997] Meij, H.V.D. The ISTE Approach to Usability Testing, IEEE Transactions on Professional Communication, Vol. 40, No. 3, September 1997.
48. [Musen 19889] M. Musen An Editor for the conceptual models of interactive knowledge-acquisition tools. International Journal of Man-Machine Studies 31, 673-698.
49. [Nielsen 1993] Nielsen, J. *Usability engineering*. San Fransisco, CA: Morgan Kaufmann. ISBN 0-12-518406-9.
50. [Nielsen 1994] Nielsen, J. (1994). Heuristic evaluation. In J. Nielsen and R. L. Mack (Eds.), *Usability inspection methods* (pp. 25-62). New York: John Wiley.
51. [Nigel 2000] Nigel Bevan- ISO and Industry standard for user centered design a report for Serco, UK at 2000.
52. [Noy 2000] Noy, N.F. Grosso, W. & Musen, M.A., Knowledge-Acquisition Interfaces for Domain Experts: An Empirical Evaluation of Protege-2000. Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE2000), Chicago, IL, 2000.
53. PCKAK4- Knowledge Acquisition tools (<http://www.epistemics.co.uk/Notes/55-0-0.htm>)
54. Protégé 2000 user's Guide (<http://protege.stanford.edu/useit.html>) developed by Stanford Medical Informatics at the Stanford University School of Medicine
55. [Preece 1994] Preece, J. et al. *Human-Computer Interaction*, Addison-Wesley Publication, 1994.
56. [Pureta 1994] Puerta, A. R.; Eriksson, H.; Gennari, J.; & Musen, M. A. *Model-Based Generation of User Interfaces*. Seattle WA, 1994.
57. *Performance Measurement Hand Book –version 3*, Serco Usability Services
58. [Rubin 1994] Rubin, J. “ Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests” –John Wiley and Sons, Inc. 1994.
59. [Rosson 2002] Rosson, M.B., & Carroll, J.M. 2002 "Usability Engineering: Scenario-Based Development of Human-Computer Interaction", Morgan Kaufmann Publishers.

60. [Roberts] Roberts, T.L., and Moran, T.P. The evaluation of text editors: methodology and empirical results. *Communication of ACM* 26, 4.265-283.
61. [Speel 1999] Speel P.H. et al. 1999 "Knowledge Mapping for Industrial Purposes" 12th Conference Knowledge Acquisition, Modeling and Management Voyager Inn, Banff, Alberta, Canada.
62. [Seffah et al. 2002] An integrated framework for usability measurement. 12th Internal Conference on Software Quality, Ottawa, Ontario.
63. [Shreiber 1994] "CommonKADS: A Comprehensive Methodology for KBS Development", *IEEE Expert, Intelligent Systems and their Applications*, vol. 9, no. 6, December 1994, pp.28-37.
64. [Shakel 1991] Shakel, B. Usability- context, framework, design and evaluation, *Human factors for Informatics Usability*. Cambridge University Press, Cambridge 21-38.
65. [Shneiderman 1987] Shneiderman, B. *Designing the user interface: Strategies for effective human computer interaction*. Reading, MA: Addison Wesley.
66. [Tallis, et al. 1999] "Studies of Knowledge Acquisition tools: Methodology and Lessons Learned" KAW-99 proceedings, Alberta, Canada.
67. [Telford 1999] Telford, D.G., and Brown, M. Task-based software testing. *Software Tech News* Vol. 3, No. 3, 1999.
68. Usability- Special Interest Group (<http://www.stcsig.org/usability/>)
69. [Victoria 1999] Victoria, E.L, "User Interface design and Usability Testing: An Application", A Masters thesis submitted to the School of Information and Library Science, University of North Carolina at Chapel Hill.
70. [Wharton 1994] Wharton, C., Rieman, J., Lewis, C., and Polson, P. (1994). The Cognitive Walkthrough Method: A Practitioner's Guide. In *Usability Inspection Methods*, J. Nielsen and R.L. Mack (Eds.), New York: John Wiley & Sons, pp.105-141.
71. [Wixon 1994] Wixon, D, Jones, S., Tse. L., and Casaday, J. Inspections and design reviews: Framework, history and reflection. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection methods*. John Willy & Sons, Boston, MA. 89-92.

72. Zang, Z. “ An Overview of Usability evaluation methods”
(www.cs.umd.edu/~zzi/Usabilityhome.html).

APPENDIX

APPENDIX A

A1. Please rate your degree of skill in using the following software.

	Not Applicable (0)	Not Comfortable (1)	Somewhat Comfortable (2)	Comfortable (3)	Very Comfortable (4)
Windows 95/Me/MP					
Linux/Unix					
Microsoft Word					
Microsoft Access					
Microsoft Excel					
Programming					
Others (specify)					

A2. Orientation Script

Hi, my name is Lokman. I will be working with you in today's session. Let me explain why we've asked you to come here today.

We're here to test knowledge acquisition tool Protégé 2000 and PCPAK4; in this regard we'd like your help.

You will be performing some typical tasks with these tools today; I would like you to perform as normally would. Usually, each task takes 3-5 minutes. Since, we are not measuring the task time; you may take as much time as you need. I hope you will be done within 15 minutes. You may ask me question any time, I am not answer them because this is a study. Since there is online support document we will see how helpful these documents for you.

During today's session I'll also be asking you to complete some forms and answers some questions. It's important that you answer truthfully. My only role here is to discover the flaws and advantages of the products from your perspective.

While you are working, I'll be sitting nearby taking some notes. The task session will be recorded by screen capturing software. Also your comments will be recorded through microphone for the purpose of discovering the misconception.

Do you have any question?

If not, then lets begin by having you sign the nondisclosure agreements and consent to

A3. Consent Form

I understand that screens capture and voice recordings will be made of my session. I grant Human computer interaction research group of computer science department, Concordia University to use these recordings for the purpose of research, and waive my right to review or inspect the tapes prior to their dissemination and distribution.

Please print name: _____

Signature: _____

Date: _____

A4. Task Scenario

Protégé KA tool

Please perform the following tasks using Protégé KA tool. The tasks need to be performed in the given order because every individual task is depended on the previous tasks. I want you to attend all the tasks and discover the flaws in the tool. In the mid of a task, if you think you wouldn't be able to complete the task, please discard the task, and start the next task by clicking into the given desktop icon for that task.

- 1) Create a project for student information system.
- 2) Create root level class "*student*" for the ontology.
- 3) Create subclasses for the root class in the given taxonomy: (First, create subclasses "*Graduate*", "*Undergraduate*" from the "*student*" class. In the second phase, create "*Masters*" and "*PhD*" class as subclass of "*Graduate*" class, after that create "*Full-time*" and "*Independent*" as subclass of "*Undergraduate*" class.

- 4) Create multiple inheritances for “*Independent*” class: “*Independent*” class should have another super class “*Graduate*”, as there are independent graduate students.
- 5) Delete the class “*Full-time*” from the hierarchy.
- 6) Create the following slots with student class: name, student-id, previous-institute, and nationality, scholarship, CGPA.
- 7) Specify the artifacts for the slots created in task 5.
- 8) Attach the slot Scholarship with the “*Graduate*” class.
- 9) Delete the “previous-institute” slot.
- 10) Customize the form for “*Graduate*” class.
- 11) Insert the following instance for student class into the knowledge base:
 - i. Name: Jasmine
 - ii. Student-id: 4355362
 - iii. Nationality: Canada
- 12) Create a query to find out the list of graduate students who are receiving scholarship.

PCPACK KA Tools

Please perform the following tasks using PCPACK KA tool. The tasks need to be performed in the given order because every individual task is depended on the previous tasks. I want you to attend all the tasks and discover the flaws in the tool. In the mid of a task, if you think you wouldn't be able to complete the task, please discard the task, and start the next task by clicking into the given task number (e.g. Task3) in the tool launcher window of PCPACK.

- 1) Create an empty knowledgebase for student information system with the ladder tool.
- 2) Create a new “*student-info*” ladder

- 3) Create root node "*Student*" for the new ladder.
- 4) Create the following children nodes: (First, create node "*Graduate*", "*Undergraduate*" as the direct children of "*student*" node. In the second phase, create "*Masters*" and "*PhD*" node as direct children of "*Graduate*" node, after that create "*Full-time*" and "*Independent*" node as direct children of "*Undergraduate*" node.
- 5) Create the following attributes with the student node: name, student-id, previous-institute, nationality, and CGPA.
- 6) Create multiple inheritances for "*Independent*" node. "*Independent*" node should have another parent "*Graduate*", as there are also independent graduate students.
- 7) Delete the "*full-time*" node.
- 8) Create an annotation template for "*Graduate*" node that will display the name of the children nodes and the attributes values of that node.
- 9) Insert the following instance for the independent node:
 - i. Name: Jasmine
 - ii. Student-id: 4355362
 - iii. Nationality: Canada
- 10) Create attribute matrix with the matrix tool.

A5. Post-test Questionnaire

Question 1: Are there any parts of the interface that you found confusing or difficult to understand?

Question 2: What are the worst aspects of the tool?

Please rate the following aspects of Protégé and PCPACK KA tools in the scale of 1 to 7

(7 is easy to use, 5 for somewhat easy, 3 somewhat difficult, and 1 is difficult)

Aspects of the KA tool	Protégé (Mean satisfaction Score.)	PCPACK (Mean satisfaction score)
Starting the tool		
Creating ontology/ladder		
Customizing/ annotating the tool		
Inserting the instance		
Selecting icon from the toolbar		
Using Menu bar and popup menu		
Understanding naming/ labeling		
Overall		

APPENDIX B

B1. Participants Computer Expertise

(Scale: 0- Not Applicable, 1- Not Comfortable, 2-Somewhat Comfortable, 3-Comfortable, and 4-Very Comfortable)

Question	Value	Frequency	Percentage
Degree of comfort with Windows 95/98	0	0	0
	1	0	0
	2	0	0
	3	3	30
	4	7	70
Degree of comfort with Linux/Unix OS	0	5	50
	1	3	30
	2	2	20
	3	0	0
	4	0	0
Degree of comfort with Microsoft Excel	0	1	10
	1	1	10
	2	1	10
	3	2	20
	4	5	50
Degree of comfort with other Tools	0	4	40
	1	1	10
	2	1	10
	3	0	0
	4	4	40
Degree of comfort with Microsoft word	0	0	0
	1	0	0
	2	0	0
	3	3	30
	4	7	70
Degree of comfort with programming	0	6	60
	1	1	10
	2	0	0
	3	0	0
	4	3	30

Table: Users expertise with other applications

B2. Knowledgebase published by PCPACK4

Published Web - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address: C:\KnowledgeWeb\studentKB\frame.htm

Trees Ladders Diagrams Project information

Expand Minimize Ladder

- Student
 - Graduate
 - Masters
 - 4438566
 - 4524337
 - PhD
 - 6534276
 - 3425262
 - Independent
 - Grad-Diploma
 - Undergraduate
 - Part-time
 - Full-time

Related links

- Masters

Object Name: 4624337

- Sub types

- Super Types

- Masters

Attribute	Value	Inherited from
Name	James	
Residency	Canadian	
Scholarship	TA	

Your Company Parent Diagram Expand contents Home Search Terminology List Expand page Help

B3. Knowledgebase Browsing by Protégé Query

Protégé 2.2.1

Graduate Residency Canadian

KB_341590_instance_17

Name: Jonathon

Residency: Canadian

Residency: RA

Query Name

Open Query